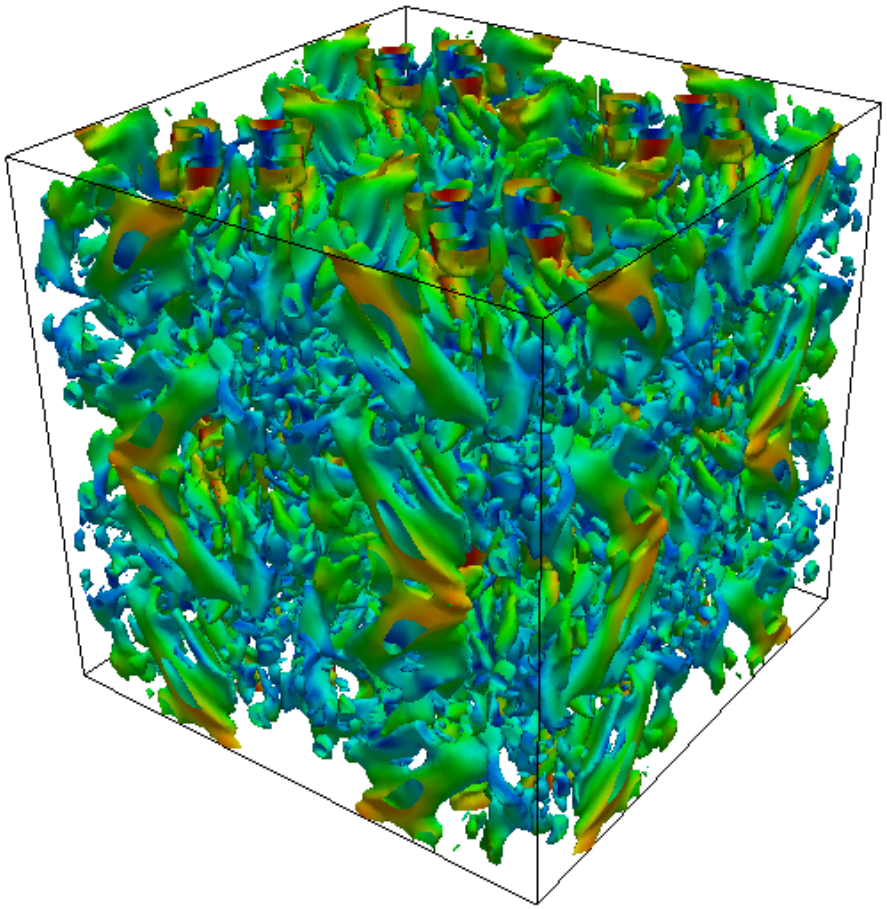
Study for the Validation of Code Saturne for Turbulent Flow Simulations

Report

Jordi Amat Foraster



UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

June 2017

Study for the Validation of Code Saturne for Turbulent Flow Simulations

Report

by

Jordi Amat Foraster

to obtain the degree of Bachelor of Science

in Aerospace Technology Engineering,

Escola Superior d'Enginyeries Industrial, Aeronàutica i Audiovisual de Terrassa (ESEIAAT)

Universitat Politècnica de Catalunya

Student name:	Jordi Amat Foraster	
Project duration:	February 15, 2017 – June 10, 2017	
Thesis supervisor/s:	Prof. Arnau Miró Jané,	ESEIAAT (DFIS), director
	Prof. Dr. Manel Soria Guerrero,	ESEIAAT (DFIS), director

On his death bed, Heisenberg is reported to have said,
"When I meet God, I am going to ask him two questions: Why relativity? And why turbulence? I really believe he will have an answer for the first."

Abstract

Nowadays, turbulent flows are of great interest in many applications because they mix fluid much more effectively than a comparable laminar flow due to its diffusivity. The conservation of the properties of the Navier-Stokes equations is of extremely importance for an accurate description of turbulent flows. This thesis analyzes the behavior of *Code_Saturne*, a multipurpose open-source CFD software package that is included in the Partnership for Advanced Computing in Europe (PRACE), in terms of kinetic energy conservation in turbulent flows. In order to do so, *Code_Saturne* is compared with a self-made spectro-consistent 2D code and another spectro-consistent 3D code that exactly preserve the symmetries of the underlying differential operators of the Navier-Stokes equations, i.e. the convective operator is approximated by a skew-symmetric matrix and the diffusive operator by a symmetric, positive-definite matrix.

The well known benchmark cases of the 2D Taylor vortex and the 3D Taylor-Green vortex are solved. A sensitivity analysis is performed in order to assess the best parameters of *Code_Saturne* that yield the best performance for both structured and unstructured meshes. The results show that *Code_Saturne* strictly conserves kinetic energy on regular Cartesian grids if an appropriate fully centered scheme is used together with the deactivation of the Rhie Chow interpolation. On 3D grids, the fully turbulent flow reveals that the presence of three dimensional effects due to vortex stretching causes a numerical dissipation that can be overcomed by refining the mesh size. However, this formulation does not allow to strictly preserve kinetic energy on unstructured grids due to the instabilities generated on the pressure gradient term. Finally, a set of configuration parameters that follow the aforementioned properties for *Code_Saturne* are provided.

Study for the Validation of Code Saturne for Turbulent Flow Simulations

by Jordi Amat Foraster

Terrassa, June 2017

Contents

List of Figures

List of Tables

Abbreviations

CFD	**C**omputational **F**luid **D**ynamics
NS	**N**avier-**S**tokes
DNS	**D**irect **N**umerical **S**imulation
LES	**L**arge-**E**ddy **S**imulation
RANS	**R**eynolds-**A**veraged **N**avier-**S**tokes
UPC	**U**niversitat **P**olitècnica de **C**atalunya
EDF	**É**lectricité **D**e **F**rance
GUI	**G**raphical **U**ser **I**nterface

Symbols

Latin

C	discrete convection operator	$kg \cdot s^{-1}$
D	discrete diffusion operator	$kg \cdot s^{-1}$
E_k	kinetic energy	J
G	discrete gradient operator	m^{-1}
k	wave length	m
l	characteristic length scale	m
L	reference length	m
L	discrete laplacian operator	m^{-2}
M	discrete divergence operator	m^{-1}
N	number of control volumes	$-$
R	rate-of-rotation tensor	s^{-1}
S	rate-of-strain tensor	s^{-1}
p	pressure	Pa
$\mathbf{p}_c$	discrete pressure vector	Pa
t	time	s
u, v, w	velocity cartesian components	$m \cdot s^{-1}$
u	velocity vector	$m \cdot s^{-1}$
$\mathbf{u}_s$	discrete velocity vector	$m \cdot s^{-1}$
V_0	reference velocity	$m \cdot s^{-1}$
x, y, z	cartesian coordinates	m
x	position vector	m

Greek

Δ	reference cell size	m
Δt	time step	s
ε	kinetic energy dissipation rate	W

η	Kolmogorov length scale	m
μ	dynamic viscosity	$Pa \cdot s$
ν	kinematic viscosity	$m^2 \cdot s^{-1}$
ρ	density	$kg \cdot m^{-3}$
ω	vorticity vector	s^{-1}
Ω	volume	m^3
Ω_s	discrete volume operator	m^3
τ	Kolmogorov time scale	s
ζ	enstrophy	$kg \cdot s^{-2}$

Dimensionless

Re	Reynolds number

Superscripts

d	deviatoric
p	predictor
t	transposed
o	isotropic
$*$	adimensional

Subscripts

0	initial value
c	cell centered
s	staggered

Introduction

Since the first half of the 20th century, when the most important aerodynamicists contributed to the field with theories that constantly changed the paradigm of the recently-born air science, few significant changes can be noticed. The reason for this stagnation is the contrast in difficulty between the characterization of laminar and turbulent flows. While laminar flows follow already known patterns and their links with the aerodynamic forces are clear and predictable, turbulence remains as a yet unsolved phenomenon.

From the late seventies on, the potential provided by the first accessible computers generated high expectations in terms of the ability to solve problems that entailed a non-feasible amount of human workload and allowed the practical growth of Computational Fluid Dynamics (CFD), which is a branch of fluid mechanics that uses numerical analysis and data structures to solve and analyze problems that involve fluid flows. The basic idea is to use appropriate algorithms to find solutions to the equations describing the fluid motion. In the aeronautics field, CFD has emerged as a new "third dimension"[4], complementing the previous dimensions of both pure experiment and pure theory. It allows the obtention of answers to fluid dynamic problems which were intractable by classical analytical methods. Consequently, CFD is revolutionizing the airplane design process, and in many ways is modifying the way as modern aeronautical research and development is conducted.

Although this computational progress has significantly helped to solve several problems regarding physics, mathematics, and engineering, the turbulence problem still remains. Unfortunately, the complexity of these cases is inherently related to the computational cost that they require and also to the software needed to solve them, which is not usually free. Therefore, advances in CFD, and its application to problems of more and more detail and sophistication, are intimately related to advances in computer hardware, particularly in regard to storage and execution speed. This is why the strongest force driving the development of new supercomputers is coming from the CFD community.

During the last decade and despite the aforementioned difficulties, many individuals and researchers have devoted their efforts to develop several free and open-source tools with enough power to solve complex problems and to serve as an accessible CFD tool. In this sense, open-source status allows for answers to specific needs that cannot easily be made available in commercial ”black box” packages. In the framework of these tools, *Code_Saturne* software package has arisen as one of the 12 solvers selected by the Partnership for Advanced Computing in Europe (PRACE) project and has been thoroughly proved to scale on large systems.

The credibility of a CFD code is obtained by demonstrating acceptable levels of uncertainty and error, which are determined through verification and validation assessment. The means to achieve highly fidelity computational simulations of fluid dynamic phenomena is analyzed by considering the interactions among the various constituent parts of the simulation hierarchy including the mathematical model of physics, the numerical model and the computational model (including the mesh).

In particular, the accuracy of a CFD tool to properly solve a turbulent flow mostly lies in the idea of having symmetry-preserving discretization schemes that avoid numerical dissipation. Therefore, to assess how *Code_Saturne* is able to characterize and solve turbulent flows this work is structured as follows. First a state of the art on the nature of turbulence, the symmetry-preserving philosophy and *Code_Saturne* software is presented in order to gain insight in the field of study as well as to check what has been done before in the line of the present study. Then, a general approach of the symmetry-preserving discretization of the Navier-Stokes equations is carried in order to program the spectro-consistent code. Finally, the results of the selected benchmark test cases are presented. The lid-driven cavity problem is simulated in *Code_Saturne* because is the best way to learn the software, whereas the 2D Taylor vortex and the 3D Taylor-Green vortex flows are simulated in *Code_Saturne* and analyzed in comparison to the spectro-consistent code to check whether the software dissipates or not in a 2D and 3D configuration, respectively. Finally, with all the extracted conclusions the best configuration for *Code_Saturne* is provided.

With all that, a research study on this subject is the completion of an academic cycle by which I intend to contribute to the assessment of *Code_Saturne* open-source accessible CFD tool so that it may be used with confidence for aerodynamic simulation.

Objectives

The main aim of this study is to assess how *Code_Saturne* is able to characterize and solve turbulent flows. To do so, several objectives have to be accomplished.

The first objective of this BSc thesis is to gain insight in *Code_Saturne* CFD software to be able to perform simulations with it. For this reason, the BSc candidate took part in some introductory seminars conducted by his thesis directors at the beginning of the project. Then, a literature study is performed in order to select which benchmark problems should be simulated to perform the analysis.

The second objective of this BSc thesis is to investigate into the symmetry-preserving conditions that a CFD code has to maintain in order to avoid numerical dissipation. Afterwards, a two dimensional spectro-consistent solver of the Navier-Stokes (NS) equations is coded following the guidelines of the thesis directors.

Finally, the third objective of this BSc thesis is to perform simulations of the selected benchmark test cases with both *Code_Saturne* and self-made code (including the pre-process). Then post process the results and carry out a complete analysis of them in comparison to benchmark data to extract the study conclusions.

Scope

The scope of this BSc thesis is the study of turbulent flows through Direct Numerical Simulation (DNS). This implies understanding the physical and mathematical models that define turbulence to postprocess significant data.

It is part of the scope to familiarize with *Code_Saturne* workflow, to program subroutines in C, Fortran, Phyton and Matlab languages to obtain the desired results and to perform simulations of benchmark problems under different configurations.

It is also part of the scope the development of a two dimensional spectro-consistent NS solver. However, it is out of the scope developing its generalization to three dimensions.

Although this thesis will deal with parallel CFD codes, it is not part of the scope their development, neither the implementation of turbulence models.

On the other hand, sequential and parallel post processing tools and their validation are part of the scope, since they are needed in order to perform visualization and statistical analysis of turbulence.

Finally, it is really important to learn how to properly manage the constraints in terms of computational time and cost.

1

State of the Art

In this chapter, a state of the art of the fields involving the present study is presented. Firstly, the nature of turbulent flows and its main modeling methodologies are summarized. Following, the idea behind the symmetry-preserving methods is presented. Finally, some of the main features and relevance of *Code_Saturne* are outlined.

1.1 Turbulence

In engineering applications turbulent flows are prevalent, but less easily seen. In the processing of liquids or gases with pumps, compressors, pipe lines, etc., the flows are generally turbulent. Similarly the flows around vehicles (e.g., airplanes, automobiles, ships, and submarines) are turbulent. The mixing of fuel and air in engines, boilers, and furnaces, and the mixing of the reactants in chemical reactors take place in turbulent flows.[1]

An important characteristic of turbulence is its diffusivity, the ability to transport and mix fluid much more effectively than a comparable laminar flow. The effectiveness of turbulence for transporting and mixing fluids is of prime importance in many applications. When different fluid streams are brought together to mix, it is generally desirable for this

mixing to take place as rapidly as possible. Turbulence is also effective at mixing the momentum of the fluid. As a consequence, on aircraft's wings and ships' hulls the wall shear stress (and hence the drag) is much larger than it would be if the flow were laminar. Similarly, compared with laminar flow, rates of heat and mass transfer at solid-fluid and liquid-gas interfaces are much enhanced in turbulent flows.[1]

Taking the above into account, the motivation for the study of turbulent flows is clear. However, the understanding and prediction of turbulent fluid flow continues to be one of the pacing items in the physical sciences because it requires calculating turbulent velocity fields which are three-dimensional, time-dependent and chaotic.

In this sense, Richardson tried to explain the behavior of turbulent flow through the theory of energy cascade[5] where the flow is composed by "eddies" (structures) of different sizes. In brief, the idea of the energy cascade is that kinetic energy enters the turbulence through the production mechanism at the largest scales of motion. This energy is then transferred by inviscid processes to smaller and smaller scales until, at the smallest scales, the kinetic energy is dissipated into internal by viscous action.

In his original theory of 1941, Kolmogorov tried to quantify this cascade through dimensional analysis. He first postulated that for very high Reynolds numbers, the small scale turbulent motions are statistically isotropic[6]. In general, the large scales of a flow are not isotropic, since they are determined by the particular geometrical features of the boundaries (the size characterizing the large scales can be denoted as L). Kolmogorov's idea was that in the Richardson's energy cascade this geometrical and directional information is lost, while the scale is reduced, so that the statistics of the small scales has a universal character. Thus, Kolmogorov introduced a second hypothesis: for very high Reynolds numbers the statistics of small scales are universally and uniquely determined by the kinematic viscosity ν and the rate of energy dissipation ε[6]. With only these two parameters, the unique length and time scales that can be formed by dimensional analysis[1] are the so called Kolmogorov scales

$$\eta = \left(\frac{\nu^3}{\varepsilon}\right)^{1/4} \qquad \tau = \left(\frac{\nu}{\varepsilon}\right)^{1/2} \tag{1.1}$$

Dissipation of kinetic energy takes place at scales of the order of Kolmogorov length η, while the input of energy into the cascade comes from the decay of the large scales, of order L. In between of these two scales at the extremes of the cascade there is a range of scales, each one with its own characteristic length l, where inertial effects are still much larger than viscous effects, and it is possible to assume that viscosity does not play a role in their internal dynamics (for this reason this range is called ”inertial range”).

Hence, the third hypothesis of Kolmogorov was that at very high Reynolds number the statistics of scales in the range $\eta \ll l \ll L$ are universally and uniquely determined by the scale l and the rate of energy dissipation ε[6].

The way in which the kinetic energy is distributed over the multiplicity of scales is a fundamental characterization of a turbulent flow. This is usually done by means of the energy spectrum function E_k, where k is the wavelength corresponding to some harmonics in a Fourier representation of the flow velocity field:

$$k = \frac{2\pi}{l} \tag{1.2}$$

Therefore, by dimensional analysis, the only possible form for the energy spectrum function[6] according with the third Kolmogorov's hypothesis is

$$E_k \sim \varepsilon^{2/3} k^{-5/3} \tag{1.3}$$

and the ratio between largest and the smallest scales[1] are

$$\frac{l}{\eta} \sim Re^{4/3} \quad \frac{t}{\tau} \sim Re^{1/2} \tag{1.4}$$

This is one of the most famous results of Kolmogorov 1941 theory, and considerable experimental evidence[6] supports it.

The idea of energy cascade is the starting point to understand the several closures that exist for turbulent flows. The three main modeling methodologies are briefly described below and classified in Fig. 1.1.

1.1.1 Direct Numerical Simulation

Direct Numerical Simulation (DNS) is the direct resolution of the unsteady Navier-Stokes (NS) equations which describe the physics of turbulent flows without the need of any additional model. It allows to resolve all the relevant scales of a turbulent flow, providing spatially resolved turbulence statistics and second order statistics such as Reynolds stress tensor, turbulent kinetic energy, its generation and dissipation. However, because the convective term produces far too many dynamically relevant scales of motion DNS is computationally expensive.

Taking into account Eq. 1.4 it follows that to describe a flow accurately in a numerical simulation on a uniform grid, the minimum number of grid points per (integral scale)^3 is $\sim Re^{9/4}$. One consequence of this is that DNS storage requirements grow at least with $\sim Re^{9/4}$ and since the time step has usually to be taken proportional to the spatial mesh, the computational cost grow at least with $\sim Re^3$ according to Frisch[6]. Despite the advances in computer hardware, it is too high for many flows of technical interest and, in practice DNS has to be restricted to low Reynolds numbers.

1.1.2 Large-Eddy Simulation

Large-Eddy Simulation (LES)[7] is based on a spatial averaging (low pass filtering) of the NS equations. LES directly computes the large-scale turbulent structures, those that contain most of the turbulent kinetic energy, and therefore are still very expensive in terms of CPU cost. However, the rest of the scales, down to the Kolmogorov scale η (see Fig. 1.1) are modeled with a sub-grid scale model to capture the effects of the smaller unresolved scales, minimizing the model's influence.

The original and dynamic Smagorinsky models, based on the use of a linear eddy viscosity model are still widely used. For very fine meshes, the effect of the model vanishes and the original NS equations are recovered. The Wall-Adapting Local Eddy-viscosity model (WALE)[8] allows to solve the difficulties associated with the application of LES to complex flows and geometries.

1.1.3 Reynolds-Averaged Navier-Stokes

Reynolds-Averaged Navier-Stokes (RANS), based on time-averaging the NS equations, are still in wide use and are the most cost-effective method of computing turbulent flows. The idea behind the equations is Reynolds decomposition, whereby an instantaneous quantity is decomposed into its time-averaged and fluctuating quantities. The mean velocity fields are solved, while the nonlinear Reynolds stresses are modeled, resulting in the different variations of RANS models.

Among the most important: $k-\varepsilon$ model[9], probably the most common turbulence model used in CFD, belongs to the class of two-equations models. Turbulent kinetic energy and its dissipation are modeled. Shear Stress Transport (SST) model[10], also a two equation model, combines a $k-\omega$ model with a $k-\varepsilon$ model to deal with the strong freestream sensitivity of the $k-\omega$ model and improve the predictions of adverse pressure gradients. Spalart-Allmaras (SA) model[11] is a one-equation model originally developed for external aerodynamics that solves a modelled transport equation for the kinematic eddy turbulent viscosity. Despite its simplicity it is nowadays one of the most used turbulence models in CFD, specially in external aerodynamics.

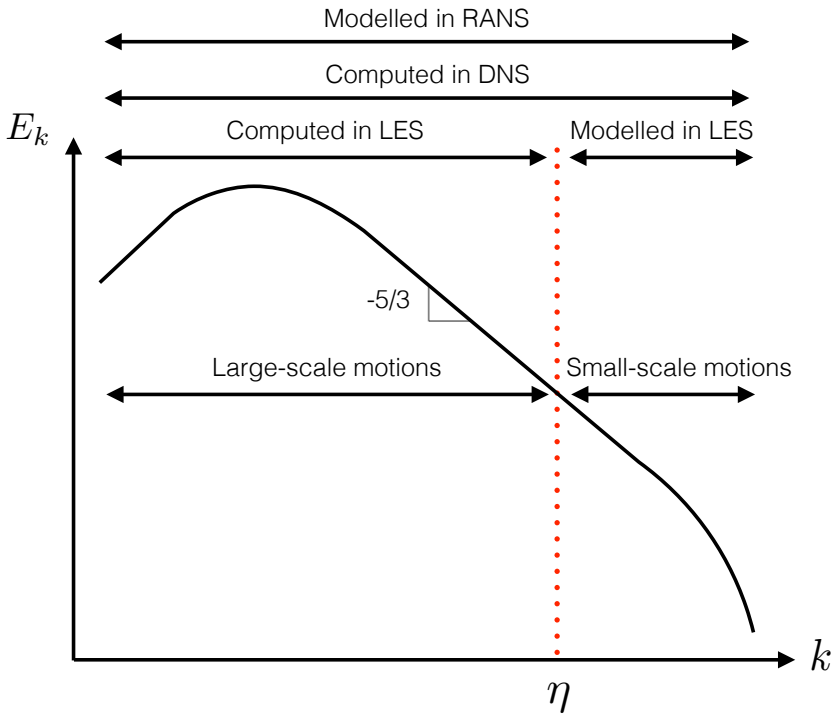


Figure 1.1: Classification of turbulence modeling methodologies in terms of scale motions (inspired by Pope[1]).

1.2 Symmetry-Preserving

Finding an equilibrium between accuracy and stability has always been a great challenge for numerical simulation of turbulence. For example, upwind-like schemes are very stable but they introduce artificial numerical dissipation that systematically damps kinetic energy, or central difference schemes do not add artificial numerical dissipation but they do not guarantee stability.

The importance of conservative discretization methods was first realized in the pioneering work by Arakawa[12], which showed that the conservation of both kinetic energy and enstrophy by convective schemes prevent systematic and unrealistic energy cascade towards high wavenumbers, a cause of nonlinear instability. This philosophy has been followed by several investigators meanwhile others have dedicated their efforts to develop high order methods, eventhough nowadays it is generally accepted that the quality of the results is not automatically improved by simply increasing the order of accuracy of the numerical scheme. Instead, the numerical schemes should retain the symmetry properties of the continuous equations.

In this sense, Verstappen and Veldman[13,14] proposed to exactly preserve the symmetry properties of the underlying differential operators because the smallest scales of motion in a turbulent flow result from a subtle balance between convective transport and diffusive dissipation. The convective operator is discretized by a skew-symmetric matrix and the diffusive operator by a symmetric, positive-definite matrix. The authors showed that such conditions are enough to ensure stability and proved that the second-order symmetry-preserving discretization proposed in[13,14] yields a second-order accurate solution for structured non-uniform grids using a staggered arrangement.

The way for accurate simulations on more complicated domains was opened by Perot[15], who derived a conservative staggered mesh scheme for unstructured grids. Later, Hicken *et al.*[16] presented a fully conservative method for staggered unstructured grids, however, it has only been put in practice on orthogonal unstructured grids. More recently, Trias *et al.*[17] in collaboration with Verstappen, published a fully-conservative discretization on collocated unstructured grids. Here, a novel approach to eliminate the checkerboard spurious modes without introducing any non-physical dissipation is proposed. To do so, a fully-conservative regularization of the convective term was used

and the new discretization method was successfully tested.

Finally, the author of this thesis wants to remark that very recent works in this line can be found in addition to the ones cited here and that UPC investigators (among which are this thesis' directors) have been committed to this line of research for the last decade.

1.3 *Code_Saturne*

Code_Saturne is a multipurpose open source CFD software package developed since 1997 at Électricité de France (EDF), one of the biggest electricity producer in Europe, and distributed under the GNU GPL license since 2007. It is based on an parallel unstructured collocated finite volume approach that accepts meshes with any type of cell (tetrahedral, hexahedral, prismatic, pyramidal, polyhedral...) and any type of grid structure (unstructured, block structured, hybrid, conforming or with hanging nodes...).

Code_Saturne solves the NS equations for 3D flows, steady or unsteady, laminar or turbulent, incompressible or weakly dilatable, isothermal or not, with scalars transport if required. Several turbulence models are available, from RANS models to LES models. In addition, a number of specific physical models are also available as "modules": gas, coal and heavy-fuel oil combustion, semi-transparent radiative transfer, particle-tracking with Lagrangian modeling, Joule effect, electrics arcs, weakly compressible flows, atmospheric flows, rotor/stator interaction for hydraulic machines.

Furthermore, *Code_Saturne* provides great flexibility since it can be coupled with other softwares for joint computations:

- *Syrthes* for thermal-fluid computations.
- *Code_Aster* for structure-fluid computations.
- *Code_Saturne* for fluid-fluid computations.

Code_Saturne may be installed on a Linux or other Unix like system by downloading and building it. In this sense, *Code_Saturne* installation manual[18] provides a simple guide to install it. The code is written in Fortran (49%), C (41%) and Python (10%).

It allows parallel coding using MPI and it has been used extensively and validated on high-performance computing machines such as Opteron, Itanium, Power5-6 and Blue-Gene[19]. Its great scaling properties have made *Code_Saturne* to be selected as a CFD applications benchmark to be used for assessing the performance of high-performance computing systems under the Partnership for Advanced Computing in Europe (PRACE) project[20].

The code was originally designed for industrial applications and research activities in several fields related to energy production; typical examples include nuclear power thermal-hydraulics, gas and coal combustion, turbo-machinery, heating, ventilation, and air conditioning. However, since EDF released the code as open-source it provided both industry and academia to benefit from its extensive pedigree as *Code_Saturne*'s open-source status allows for answers to specific needs that cannot easily be made available in commercial "black box" packages.

The software has been constantly under development and updated by EDF in collaboration with several research centers. *Code_Saturne* 1.0 was validated and released early in 2000 providing basic modelling and wide range of meshes. Efforts then were invested during one year into a validation process for nuclear single-phase thermal-hydraulics until it was qualified[21]. In 2004, the *Code_Saturne*'s numerical method was presented in Archambeau *et al.*[22].

Since then, great efforts have been dedicated to validate its numerical method to compute turbulent incompressible flows and many researchers have been qualifying the several RANS and LES turbulence models available in *Code_Saturne*. Among the others, Dr. Benhamadouche's PhD thesis[23] is of great interest for the purpose of the present thesis since a deep analysis was performed into the accuracy and stability of LES simulations on unstructured grids using the collocated arrangement employed in *Code_Saturne*. He realized that conserving kinetic energy is important in LES so that sub-grid-scale modelling is not hidden by numerical errors. Thus, he followed the symmetry-preserving philosophy presented in Section 1.2 to analyze the energy conservation of *Code_Saturne*. In his conclusions, he encountered two main problems:

1. In the second order unstructured formulation, the pressure gradient and the second order convection scheme did not allow to conserve energy on non-orthogonal grids. No alternative solution was found for the pressure gradient. The convection scheme could conserve global kinetic energy with a symmetrical formulation, but this scheme altered the precision of the numerical scheme to first order on non-uniform meshes.

2. The Rhie and Chow interpolation[24] used in all the collocated approaches introduced a numerical dissipation which was particularly noticeable in inviscid flows. However, removing Rhie and Chow interpolation could entail unstable solutions on fully unstructured meshes.

Despite that, he concluded that at least removing this interpolation for regular Cartesian grids should allow, with a second order Crank-Nicolson scheme and sub-iterations on the predictor-corrector algorithm for pressure velocity coupling, to strictly conserve kinetic energy.

The present work will follow this problematic in order to quantify how *Code_Saturne* dissipates energy under different configurations. Then, based on the results it will provide the most suitable configuration to avoid dissipation in DNS simulations.

Finally, the author of this thesis recommends any reader interested in learning how to run a simulation in *Code_Saturne* to refer to Appendix A, which provides a simple guide for beginners to *Code_Saturne*.

2

Numerical Discretization of the Navier-Stokes equations

In this chapter, a description of the equations that govern fluid motion and its numerical discretization to solve them are presented.

2.1 Governing Equations

In this section, a review of the equations that govern fluid motion is performed to better understand the mathematical model of fluid dynamics.

2.1.1 Incompressible Navier-Stokes Equations

Both laminar and turbulent flows of constant-property Newtonian fluids at low Mach numbers are governed by the incompressible Navier-Stokes (NS) equations, which describe the motion of a fluid in a bounded domain Ω of $\mathbb{R}^n$ (n = 2 or 3). Since fluid density and viscosity depend on the temperature but both are supposed constant, one does not need to compute the temperature field as it plays no role in the fluid dynamics.

The incompressible NS equations are given by a continuity (scalar) equation and a momentum (vectorial) equation

$$\nabla \cdot \mathbf{u} = 0 \tag{2.1}$$

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = \nu \Delta \mathbf{u} - \frac{1}{\rho}\nabla p \tag{2.2}$$

where t is the time, **u** is the velocity field, p is the pressure, ν is the kinematic viscosity and ρ is the density. The first term of Eq. 2.2 represents the accumulative term, the second is the convective term, the third represents the diffusive term and the fourth term contains the pressure contribution.

The dimensionless NS equations can be easily derived as

$$\nabla^* \cdot \mathbf{u}^* = 0 \tag{2.3}$$

$$\frac{\partial \mathbf{u}^*}{\partial t^*} + (\mathbf{u}^* \cdot \nabla^*)\mathbf{u}^* = \frac{1}{Re}\Delta^* \mathbf{u}^* - \nabla^* p^* \tag{2.4}$$

where the dimensionless magnitudes (denoted by *) are obtained through the characteristic velocity V_0 and length L

$$t^* = \frac{t}{L/V_0} \quad \mathbf{x}^* = \frac{\mathbf{x}}{L} \quad \mathbf{u}^* = \frac{\mathbf{u}}{V_0} \quad p^* = \frac{p}{\rho V_0^2}$$

Re is the dimensionless Reynolds number defined as

$$Re = \frac{\rho V_0 L}{\mu} = \frac{V_0 L}{\nu} \tag{2.5}$$

The importance of Reynolds number comes from Eq. 2.4, which shows that at high flow rates the diffusive term becomes much smaller compared to the convective term because with increasing flow rates, the inertial forces dominate over the viscous forces.

2.1.2 Pressure Poisson Equation

The pressure Poisson equation can be obtained by taking the divergence of Eq. 2.2

$$\nabla \cdot \left(\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} \right) = \nabla \cdot \left(\nu \Delta \mathbf{u} - \frac{1}{\rho}\nabla p \right) \tag{2.6}$$

The latter expression can be simplified with some algebra, reducing equation to

$$\frac{\partial}{\partial t}(\nabla \cdot \mathbf{u}) + \nabla \cdot (\mathbf{u} \cdot \nabla)\mathbf{u} = \nu \Delta (\nabla \cdot \mathbf{u}) - \frac{1}{\rho}\Delta p \tag{2.7}$$

Considering an incompressible fluid (Eq. 2.1), the pressure Poisson equation is then obtained:

$$\nabla \mathbf{u} : \nabla \mathbf{u} = -\frac{1}{\rho}\Delta p \tag{2.8}$$

This equation shows that for incompressible flows, there is no connection between pressure and density, and a different understanding of pressure is required. In consequence, local velocity perturbations can be propagated instantaneously to all the domain through pressure.

2.1.3 Vorticity Transport Equation

The vorticity transport equation can be obtained by taking the curl of Eq. 2.2

$$\frac{\partial \omega}{\partial t} + \nabla \times ((\mathbf{u} \cdot \nabla)\mathbf{u}) = \nu \nabla \times \Delta \mathbf{u} - \frac{1}{\rho}\nabla \times \nabla p \tag{2.9}$$

where $\omega = \nabla \times \mathbf{u}$ is the vorticity. The latter expression can be simplified with some algebra, reducing the vorticity equation to

$$\frac{\partial \omega}{\partial t} + \nabla \times (\omega \times \mathbf{u}) = \nu \Delta \omega \tag{2.10}$$

Considering an incompressible fluid (Eq. 2.1), the vorticity transport equation is then obtained:

$$\frac{\partial \omega}{\partial t} + (\mathbf{u} \cdot \nabla)\omega = \nu \Delta \omega + (\omega \cdot \nabla)\mathbf{u} \tag{2.11}$$

where the first term represents the temporal evolution of the vorticity, the second is the convective term, the third represents the viscous effects and the fourth is the vortex-stretching term.

It can be proved that the vortex-stretching vector is related to the symmetric deviatoric rate-of-strain tensor $\mathbf{S} = \frac{1}{2}\left(\nabla \mathbf{u} + \nabla \mathbf{u}^t\right)$ by

$$(\omega \cdot \nabla)\mathbf{u} = \frac{1}{2}\left(\nabla \mathbf{u} + \nabla \mathbf{u}^t\right)\omega = \mathbf{S}\omega \tag{2.12}$$

This tensor is symmetric and traceless as described in Appendix B.

$$tr(\mathbf{S}) = \nabla \cdot \mathbf{u} = 0 \tag{2.13}$$

Since the trace is null (Eq. 2.13) it implies that, at least, one eigenvalue of **S** is positive because the trace of a symmetric matrix is the sum of its eigenvalues. Hence, if vorticity, ω, aligns with an eigenvector of **S** corresponding to a positive eigenvalue then the term $(\omega \cdot \nabla)\mathbf{u}$ becomes positive and amplifies (by stretching) the vorticity.[25]

Note that for two-dimensional flows, the vortex-stretching term vanishes ($\omega \cdot \nabla \to 0$), and the one non-zero component of vorticity evolves as a conserved scalar. Because of the absence of vortex-stretching, two-dimensional turbulence (which can occur in special circumstances) is qualitatively different than three-dimensional turbulence.[1]

2.1.4 Global Kinetic Energy Equation

The integrated kinetic energy within the domain Ω is computed by

$$E_k = \frac{1}{\rho_0 \Omega} \int_\Omega \rho \frac{\mathbf{u} \cdot \mathbf{u}}{2} \, d\Omega \tag{2.14}$$

The kinetic energy dissipation rate can then be computed by differencing E_k in time. In Appendix C it is proofed that in the absence of external sources (such as body or boundary forces) the rate of change of the total energy is neither influenced by convective transport nor by pressure differences; it is solely determined by viscous dissipation for an incompressible flow. Hence, the energy equation reduces to

$$\varepsilon = -\frac{dE_k}{dt} = \frac{2\nu}{\Omega} \int_\Omega \frac{\nabla \mathbf{u} \cdot \nabla \mathbf{u}}{2} \, d\Omega = \frac{2\nu}{\Omega} \int_\Omega \mathbf{S} : \mathbf{S} \, d\Omega \tag{2.15}$$

In addition, it can be shown in Appendix C that for an incompressible flow, the enstrophy is directly related to the kinetic energy dissipation rate through a constant.

$$\varepsilon(\zeta) = -\frac{dE_k}{dt} = \frac{2\nu}{\Omega} \int_\Omega \frac{\omega \cdot \omega}{2} \, d\Omega = 2\nu\zeta \tag{2.16}$$

where the enstrophy integrated on the domain Ω is

$$\zeta = \frac{1}{\rho_0 \Omega} \int_\Omega \rho \frac{\omega \cdot \omega}{2} \, d\Omega \tag{2.17}$$

2.2 Discrete Navier-Stokes Equations

The finite volume discretization of the NS equations (Eq. 2.1 and Eq. 2.2) on an arbitrary mesh with N control volumes can be written following a similar notation to Verstappen *et al.*[13] by

$$\mathbf{M}\mathbf{u}_s = \mathbf{0}_c \tag{2.18}$$

$$\rho\Omega_s \frac{d\mathbf{u}_s}{dt} + \mathbf{C}(\mathbf{u}_s)\mathbf{u}_s + \mathbf{D}\mathbf{u}_s + \Omega_s \mathbf{G}\mathbf{p}_c = \mathbf{0}_s \tag{2.19}$$

where:

- $\mathbf{p}_c$ is a N component vector with the discrete approximation of pressure.
- $\mathbf{u}_s$ is a $3N$ component discrete vector containing the three components of velocity.
- Ω_s is a diagonal $3N \times 3N$ matrix representing the sizes of the control volumes associated with the discrete velocity field.
- **C** is a $3N \times 3N$ matrix, function of the velocity field, that expresses the discrete convective operator (non-linear operator).
- **D** is a $3N \times 3N$ constant matrix that expresses the discrete diffusive operator.
- **G** is a $3N \times N$ constant matrix that expresses the discrete gradient operator, that is that multiplied by a scalar field yields to a vector field.
- **M** is a $N \times 3N$ constant matrix that expresses the discrete divergence operator.

The conservative nature of the NS equations is intimately tied up with the symmetries of the differential operators (see Verstappen *et al.*[13], for instance). The following subsection shows that retaining the symmetry properties of the continuous operators when discretizing equations is necessary in order to exactly conserve the inviscid invariants in a discrete sense.

2.2.1 Kinetic Energy Conservation

The total discrete kinetic energy can be calculated as:

$$|\mathbf{u}_s|^2 = \mathbf{u}_s^t \Omega_s \mathbf{u}_s \tag{2.20}$$

Its time derivative can be obtained following the same procedure shown in Appendix C but now substituing the discrete momentum equation (Eq. 2.19):

$$\frac{d}{dt}[\mathbf{u}_s^t \Omega_s \mathbf{u}_s] = -\overbrace{\mathbf{u}_s^t(\mathbf{C}+\mathbf{C}^t)\mathbf{u}_s}^{=0} - \overbrace{\mathbf{u}_s^t(\mathbf{D}+\mathbf{D}^t)\mathbf{u}_s}^{\geq 0} - \overbrace{(\mathbf{u}_s^t \Omega_s \mathbf{G}\mathbf{p}_c + \mathbf{p}_c^t \mathbf{G}^t \Omega_s^t \mathbf{u}_s)}^{=0} \tag{2.21}$$

Again, in the absence of external sources, the overall contribution from convection has to be zero, as convection only transports kinetic energy and changes the scales of the structures, but does not dissipate energy[1]. Moreover, the pressure gradient contribution has to be zero for an incompressible flow since the body forces has no effect on the velocity field and consequently on the pressure field. These conservation properties are held, if and only if, the discrete convective operator is skew-symmetric and if the negative conjugate transpose of the discrete gradient operator is exactly equal to the divergence operator[13]:

$$\mathbf{C} = -\mathbf{C}^t \tag{2.22}$$

$$\Omega_s \mathbf{G} = -\mathbf{M}^t \tag{2.23}$$

Therefore, if the convective and gradient operators are properly chosen, the global kinetic energy equation Eq. 2.21 reduces to

$$\frac{d}{dt}[\mathbf{u}_s^t \Omega_s \mathbf{u}_s] = -\mathbf{u}_s^t(\mathbf{D}+\mathbf{D}^t)\mathbf{u}_s \leq 0 \tag{2.24}$$

where the inequality follows from the condition that diffusive terms must be strictly dissipative, otherwise it would be against the Second Principle. The latter condition is satisfied if, and only if, the diffusive operator, **D**, is symmetric and positive definite[13].

So, in conclusion, for inviscid flow the energy is conserved, whereas for viscous flow the energy does not increase in time.

2.3 Simmetry-Preserving Spatial Discretization

In the previous section, it is seen that conservation properties and stability are directly related to the symmetry of the underlying differential operators. This section shows how to work this out for the incompressible NS equations, restricting the approach to two spatial dimensions. Nonetheless, the extension to 3D is straightforward.

On a uniform grid, the traditional aim is to minimize the local truncation error without breaking the symmetries of the convective and diffusive operators in the NS equations. The well-known staggered arrangement of the discrete variables of Harlow *et al.*[26] forms an example of this. The staggering of the variables leads to a method that conserves mass, momentum, energy and vorticity, and strongly couples pressure and velocity, making it the method of choice for simulating incompressible flows on cartesian grids. The setting is illustrated in Fig. 2.1, where the pressure and the velocity components are defined at different locations.

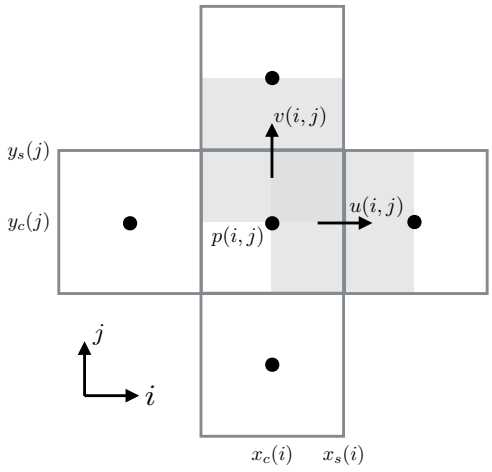


Figure 2.1: Staggered arrangement of discrete variables.

In this 2D arrangement one can distinguish the following domains:

- Cell center: $\Omega_c(i,j) = [x_s(i-1), x_s(i)] \times [y_s(j-1), y_s(j)]$
- Staggered u-component: $\Omega_{sx}(i,j) = [x_c(i), x_c(i+1)] \times [y_s(j-1), y_s(j)]$
- Staggered v-component: $\Omega_{sy}(i,j) = [x_s(i-1), x_s(i)] \times [y_c(j), y_c(j+1)]$

The NS equations (Eq. 2.1 and Eq. 2.2) integrated in this domain at a particular instant of time result

$$\int_{\Omega_c} \nabla \cdot \mathbf{u}\, d\Omega = 0 \tag{2.25}$$

$$\int_{\Omega_{sx}} \rho \frac{\partial u}{\partial t}\, d\Omega + \int_{\Omega_{sx}} \nabla \cdot (\rho u)\mathbf{u}\, d\Omega - \int_{\Omega_{sx}} \nabla \cdot (\mu \nabla u)\, d\Omega + \int_{\Omega_{sx}} \nabla p_x\, d\Omega = 0 \tag{2.26}$$

$$\int_{\Omega_{sy}} \rho \frac{\partial v}{\partial t}\, d\Omega + \int_{\Omega_{sy}} \nabla \cdot (\rho v)\mathbf{u}\, d\Omega - \int_{\Omega_{sy}} \nabla \cdot (\mu \nabla v)\, d\Omega + \int_{\Omega_{sy}} \nabla p_y\, d\Omega = 0 \tag{2.27}$$

Note that the diffusive and pressure terms have been placed in the LHS of the momentum equation according to Eq. 2.19. Now, applying the divergence theorem this system of equations becomes

$$\int_{\delta\Omega_c} \mathbf{u} \cdot \mathbf{n}\, d\Gamma = 0 \tag{2.28}$$

$$\int_{\Omega_{sx}} \rho \frac{\partial u}{\partial t}\, d\Omega + \int_{\delta\Omega_{sx}} (\rho u)\mathbf{u} \cdot \mathbf{n}\, d\Gamma - \int_{\delta\Omega_{sx}} (\mu \nabla u) \cdot \mathbf{n}\, d\Gamma + \int_{\Omega_{sx}} \nabla p_x\, d\Omega = 0 \tag{2.29}$$

$$\int_{\Omega_{sy}} \rho \frac{\partial v}{\partial t}\, d\Omega + \int_{\delta\Omega_{sy}} (\rho v)\mathbf{u} \cdot \mathbf{n}\, d\Gamma - \int_{\delta\Omega_{sy}} (\mu \nabla v) \cdot \mathbf{n}\, d\Gamma + \int_{\Omega_{sy}} \nabla p_y\, d\Omega = 0 \tag{2.30}$$

where the unit vector $\mathbf{n}$ denotes the outward normal on the surface $\delta\Omega$ of Ω.

Each term is discretized for a uniform (also valid for a non-uniform) mesh through a second order approach in the following subsections.

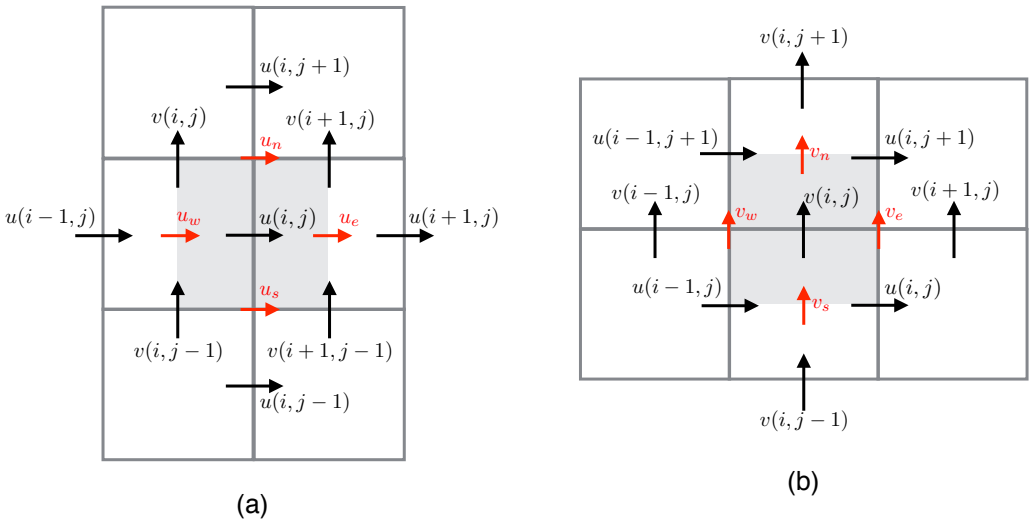


Figure 2.2: Staggered control volumes Ω_{sx} and Ω_{sy} for the discrete velocities $u(i,j)$ and $v(i,j)$, respectively.

2.3.1 Divergence Discretization

The divergence operator **M** is obtained discretizing the continuity equation:

$$\begin{aligned}(\mathbf{M}\mathbf{u}_s)_{i,j} &= \int_{\delta\Omega_c} \mathbf{u}\cdot\mathbf{n}\, d\Gamma = \int_e u\, d\Gamma - \int_w u\, d\Gamma + \int_n v\, d\Gamma - \int_s v\, d\Gamma \\ &= u(i,j)\Delta y - u(i-1,j)\Delta y + v(i,j)\Delta x - v(i,j-1)\Delta x \end{aligned} \tag{2.31}$$

where $\Delta x = x_s(i) - x_s(i-1)$ and $\Delta y = y_s(j) - y_s(j-1)$.

2.3.2 Convective Discretization

The convection operator **C** is obtained discretizing the convective term from the momentum equation.

The convective discretization for the u-component is given by

$$\begin{aligned}(\mathbf{C}_x\mathbf{u}_s)_{i,j} &= \int_{\delta\Omega_{sx}} (\rho u)\mathbf{u}\cdot\mathbf{n}\, d\Gamma = \int_e \rho u u\, d\Gamma - \int_w \rho u u\, d\Gamma + \int_n \rho u v\, d\Gamma - \int_s \rho u v\, d\Gamma \\ &= F_e u_e - F_w u_w + F_n u_n - F_s u_s \end{aligned} \tag{2.32}$$

where $F_x = \rho u_x \Delta_x$ terms are mass flows. Now, the key issue is how to evaluate the terms that are not directly available.

The velocity u_x at a control face is approximated by the average of the velocity at both sides of it (see Fig. 2.2a):

$$\begin{aligned} u_e &= \frac{u(i+1,j)+u(i,j)}{2} \quad u_w = \frac{u(i,j)+u(i-1,j)}{2} \\ u_n &= \frac{u(i,j+1)+u(i,j)}{2} \quad u_s = \frac{u(i,j-1)+u(i,j)}{2} \end{aligned} \tag{2.33}$$

Substituting the mid-point interpolation and re-ordering the terms,

$$\begin{aligned}(\mathbf{C}_x\mathbf{u}_s)_{i,j} = \frac{F_e - F_w + F_n - F_s}{2}u(i,j) + \frac{F_e}{2}u(i+1,j) - \frac{F_w}{2}u(i-1,j) \\ + \frac{F_n}{2}u(i,j+1) - \frac{F_s}{2}u(i,j-1) \end{aligned} \tag{2.34}$$

The mass flow rate terms are evaluated at their natural locations and then (if needed) they are interpolated using a mid-point rule (see Fig. 2.2a):

$$F_e = \rho\frac{u(i+1,j)+u(i,j)}{2}\Delta y \quad F_n = \frac{\rho v(i,j)\Delta x^- + \rho v(i+1,j)\Delta x^+}{2}$$
$$F_w = \rho\frac{u(i,j)+u(i-1,j)}{2}\Delta y \quad F_s = \frac{\rho v(i,j-1)\Delta x^- + \rho v(i+1,j-1)\Delta x^+}{2} \tag{2.35}$$

where $\Delta y = y_s(j) - y_s(j-1)$, $\Delta x^- = x_s(i) - x_s(i-1)$ and $\Delta x^+ = x_s(i+1) - x_s(i)$.

In addition to the set of equations for the u-component of the velocity, there is an analogous set for the v-component given by:

$$(\mathbf{C}_y\mathbf{u}_s)_{i,j} = \frac{F_e - F_w + F_n - F_s}{2}v(i,j) + \frac{F_e}{2}v(i+1,j) - \frac{F_w}{2}v(i-1,j) + \frac{F_n}{2}v(i,j+1) - \frac{F_s}{2}v(i,j-1) \tag{2.36}$$

with the mass flow terms (see Fig. 2.2b)

$$F_e = \frac{\rho u(i,j)\Delta y^- + \rho u(i,j+1)\Delta y^+}{2} \quad F_n = \rho\frac{v(i,j+1)+v(i,j)}{2}\Delta x$$
$$F_w = \frac{\rho u(i-1,j)\Delta y^- + \rho u(i-1,j+1)\Delta y^+}{2} \quad F_s = \rho\frac{v(i,j)+v(i,j-1)}{2}\Delta x \tag{2.37}$$

where $\Delta x = x_s(i) - x_s(i-1)$, $\Delta y^- = y_s(j) - y_s(j-1)$ and $\Delta y^+ = y_s(j+1) - y_s(j)$.

With this discretization, condition (2.22) is verified in two steps:

1. The matrix with the off-diagonal elements is skew-symmetric if and only if the weights in the interpolations of the discrete velocities are taken constant.

2. The matrix with the main diagonal elements is skew-symmetric if and only if its trace is zero because they are a linear combination of the mass conservation equation in the main mesh (non-staggered) and thus, if the mass conservation equation (Eq. 2.18) is satisfied, it has to be zero.

In Verstappen *et al.*[13] it is proofed that these two conditions can only be achieved when the weight in the interpolation is taken equal to $1/2$, hence independent of the grid location.

2.3.3 Diffusive Discretization

The diffusion operator **D** is obtained discretizing the diffusive term from the momentum equation.

The diffusive discretization for the u-component is given by

$$\begin{aligned}(\mathbf{D}_x\mathbf{u}_s)_{i,j} &= -\int_{\delta\Omega_{sx}} (\mu\nabla u)\cdot\mathbf{n}\, d\Gamma \\ &= -\int_e \mu\left(\frac{\partial u}{\partial x}\right) d\Gamma + \int_w \mu\left(\frac{\partial u}{\partial x}\right) d\Gamma - \int_n \mu\left(\frac{\partial u}{\partial y}\right) d\Gamma + \int_s \mu\left(\frac{\partial u}{\partial y}\right) d\Gamma \\ &= -\mu\left(\frac{\partial u}{\partial x}\right)_e \Delta y + \mu\left(\frac{\partial u}{\partial x}\right)_w \Delta y - \mu\left(\frac{\partial u}{\partial y}\right)_n \Delta x + \mu\left(\frac{\partial u}{\partial y}\right)_s \Delta x \end{aligned} \tag{2.38}$$

Following the second order approach, the partial derivatives are approximated by central differences:

$$\left(\frac{\partial u}{\partial x}\right)_e = \frac{u(i+1,j)-u(i,j)}{\delta x_e} \quad \left(\frac{\partial u}{\partial x}\right)_w = \frac{u(i,j)-u(i-1,j)}{\delta x_w} \tag{2.39}$$

$$\left(\frac{\partial u}{\partial y}\right)_n = \frac{u(i,j+1)-u(i,j)}{\delta y_n} \quad \left(\frac{\partial u}{\partial y}\right)_s = \frac{u(i,j)-u(i,j-1)}{\delta y_s} \tag{2.40}$$

Substituting and reordering terms,

$$\begin{aligned}(\mathbf{D}_x\mathbf{u}_s)_{i,j} = (D_e + D_w + D_n + D_s)u(i,j) - D_e u(i+1,j) - D_w u(i-1,j) \\ - D_n u(i,j+1) - D_s u(i,j-1)\end{aligned} \tag{2.41}$$

where the coefficients D_x are computed as

$$D_e = \mu\frac{\Delta y}{\delta x_e} \quad D_w = \mu\frac{\Delta y}{\delta x_w} \quad D_n = \mu\frac{\Delta x}{\delta y_n} \quad D_s = \mu\frac{\Delta x}{\delta y_s} \tag{2.42}$$

with

$$\begin{aligned}\Delta x &= x_c(i+1) - x_c(i) \quad \Delta y = y_s(j) - y_s(j-1) \\ \delta x_e &= x_s(i+1) - x_s(i) \quad \delta x_w = x_s(i) - x_s(i-1) \\ \delta y_n &= y_c(j+1) - y_c(j) \quad \delta y_s = y_c(j) - y_c(j-1)\end{aligned} \tag{2.43}$$

Again, in addition to the set of equations for the u-component of the velocity, there is an analogous set for the v-component given by:

$$(\mathbf{D}_y\mathbf{u}_s)_{i,j} = (D_e + D_w + D_n + D_s)v(i,j) - D_e v(i+1,j) - D_w v(i-1,j) \\ - D_n v(i,j+1) - D_s v(i,j-1) \tag{2.44}$$

where the coefficients D_x are computed through Eq. 2.42 with

$$\begin{aligned} \Delta x &= x_s(i) - x_c(i-1) \quad \Delta y = y_c(j+1) - y_c(j) \\ \delta x_e &= x_c(i+1) - x_c(i) \quad \delta x_w = x_c(i) - x_c(i-1) \\ \delta y_n &= y_s(j+1) - y_s(j) \quad \delta y_s = y_s(j) - y_s(j-1) \end{aligned} \tag{2.45}$$

2.3.4 Gradient Discretization

The gradient operator **G** is obtained discretizing the pressure term from the momentum equation.

The gradient discretization for the u-component is given by

$$(\Omega_s \mathbf{G}_x \mathbf{p}_c)_{i,j} = \int_{\Omega_{sx}} \nabla p_x \, d\Omega = \left(\frac{\partial p}{\partial x}\right)_{i,j} \Omega_{sx} = \frac{p(i+1,j) - p(i,j)}{\delta x}\Omega_s(i,j) \tag{2.46}$$

Similarly, for the v-component,

$$(\Omega_s \mathbf{G}_y \mathbf{p}_c)_{i,j} = \frac{p(i,j+1) - p(i,j)}{\delta y}\Omega_s(i,j) \tag{2.47}$$

where $\delta x = x_c(i+1) - x_c(i)$ and $\delta y = y_c(j+1) - y_c(j)$.

2.4 Pressure-Velocity Coupling

To solve the pressure-velocity coupling for an incompressible flow, a classical fractional step projection method[27] can be implemented. This projection is derived from the Helmholtz-Hodge vector decomposition theorem, which states that an arbitrary vector $\mathbf{u}_s^p$ can be transformed into a divergence-free vector $\mathbf{u}_s$ after the addition of the gradient

of a suitable scalar field $\mathbf{p}_c$:

$$\mathbf{u}_s^p = \mathbf{u}_s + \mathbf{G}\mathbf{p}_c \tag{2.48}$$

Thus, imposing the divergence-free condition,

$$\mathbf{M}\mathbf{u}_s^p = \cancel{\mathbf{M}\mathbf{u}_s} + \mathbf{M}\mathbf{G}\mathbf{p}_c \tag{2.49}$$

the linear Poisson equation for the field $\mathbf{p}_c$ is obtained:

$$\mathbf{L}\mathbf{p}_c = \mathbf{M}\mathbf{u}_s^p \tag{2.50}$$

where **L** is the discrete Laplacian operator, which by construction is symmetric, singular and negative definite matrix

$$\mathbf{L} = \mathbf{M}\mathbf{G} = -\mathbf{M}\Omega_s^{-1}\mathbf{M}^t \tag{2.51}$$

2.5 Time Discretization

To carry out the temporal discretization, Eq. 2.19 can be written as

$$\rho\frac{d\mathbf{u}_s}{dt} = \mathbf{r}(\mathbf{u}_s) - \mathbf{G}\mathbf{p}_c \tag{2.52}$$

where **r** is the vector

$$\mathbf{r}(\mathbf{u}_s) = -\Omega_s^{-1}(\mathbf{C}(\mathbf{u}_s)\mathbf{u}_s + \mathbf{D}\mathbf{u}_s) \tag{2.53}$$

Now, each term is treated in a different way: a central difference scheme for the time derivative, a second order Adam-Bashforth for **r** and an implicit first order for pressure gradient term.

In this way, the time-discrete system becomes

$$\rho\frac{\mathbf{u}_s^{n+1} - \mathbf{u}_s^n}{\Delta t} = \frac{3}{2}\mathbf{r}^n - \frac{1}{2}\mathbf{r}^{n-1} - \mathbf{G}\mathbf{p}_c^{n+1} \tag{2.54}$$

$$\mathbf{M}\mathbf{u}_s^{n+1} = \mathbf{0}_c \tag{2.55}$$

Here, the predictor velocity $\mathbf{u}_s^p$ (Eq. 2.48) is introduced as

$$\mathbf{u}_s^p = \mathbf{u}_s^n + \frac{\Delta t}{\rho}\left(\frac{3}{2}\mathbf{r}^n - \frac{1}{2}\mathbf{r}^{n-1}\right) \tag{2.56}$$

Note that it can be evaluated explicitly from the already known values of previous time steps (n and $n-1$). Then, the pressure at time step $n+1$ can be evaluated solving the Poisson equation (Eq. 2.50):

$$\mathbf{p}_c^{n+1} = \mathbf{L}^{-1}\left(\frac{\rho}{\Delta t}\mathbf{M}\mathbf{u}_s^p\right) \tag{2.57}$$

Finally, the unknown velocity $\mathbf{u}_s^{n+1}$ is obtained applying the gradient pressure correction to the predictor velocity:

$$\mathbf{u}_s^{n+1} = \mathbf{u}_s^p - \frac{\Delta t}{\rho}\mathbf{G}\mathbf{p}_c^{n+1} \tag{2.58}$$

Note that, for the first time step, only $\mathbf{u}_s^n$ is known but not $\mathbf{u}_s^{n-1}$. Thus, a first order Euler scheme for $\mathbf{r}$ can be applied:

$$\mathbf{u}_s^p = \mathbf{u}_s^n + \frac{\Delta t}{\rho}\mathbf{r}^n \tag{2.59}$$

2.5.1 Stability

Explicit discretization has a maximum time step for stability. Two conditions have to be verified due to the Courant-Friedrichs-Levy (CFL) condition[28]:

- For the diffusion terms,

$$\Delta t_d \leq \frac{1}{2}\text{min}\left(\frac{\Delta^2}{\nu}\right) \tag{2.60}$$

- For the convection terms,

$$\Delta t_c \leq \text{min}\left(\frac{\Delta}{|\mathbf{u}|}\right) \tag{2.61}$$

The time step is chosen as

$$\Delta t = f\text{min}(\Delta t_c, \Delta t_d) \tag{2.62}$$

where the coefficient f has to be below 0.3.

3

Benchmark Test Cases

In this chapter, the results of well known benchmark problems selected for the present study are presented. All the tests are performed through Direct Numerical Simulation (DNS) of the Navier-Stokes (NS) incompressible equations.

3.1 Data Processing

First of all, it is important to remark that all provided values in this chapter are properly non-dimensionalised through

$$t^* = \frac{t}{L/V_0} \quad x^* = \frac{x}{L} \quad u^* = \frac{u}{V_0} \quad \omega^* = \frac{\omega}{L/V_0} \quad E_k^* = \frac{E_k}{V_0^2} \quad \varepsilon^* = \frac{\varepsilon}{V_0^2/L^2}$$

because *Code_Saturne* solves the equations in dimensional form.

Thus, from now on the variables without the dimensionless superscript * are taken non-dimensional.

Since the objective of the present study is to test the conservation of the global energy in a discrete sense, a strategy has to be adopted in order to compute the energy losses due to numerical dissipation.

From Section 2.1.4, it is seen that, in absence of external sources, the kinetic energy dissipation rate is related to the enstrophy through a constant assuming incompressible flow. Therefore, there are two ways to compute the kinetic energy dissipation rate:

$$\varepsilon(E_k) = -\frac{dE_k}{dt} \qquad \varepsilon(\zeta) = 2\nu\zeta \tag{3.1}$$

The first is directly computed from the velocity field

$$E_k = \frac{1}{\Omega}\sum_{\Omega_i} \frac{\mathbf{u}_i \cdot \mathbf{u}_i}{2}\Omega_i \tag{3.2}$$

and the second one is computed from the vorticity field

$$\zeta = \frac{1}{\Omega}\sum_{\Omega_i} \frac{\omega_i \cdot \omega_i}{2}\Omega_i \tag{3.3}$$

Both ways should provide the same results in the cases with periodic boundary conditions. If this condition is not met, it means that there is an energy loss due to numerical dissipation ε_{dis}, which can be computed subtracting both rates:

$$\varepsilon_{dis} = \varepsilon(E_k) - \varepsilon(\zeta) \tag{3.4}$$

In the following sections, this difference will be examined to analyse the conservation properties of *Code_Saturne*.

3.2 Two-Dimensional Flow: Lid-Driven Cavity

Numerical methods for 2D incompressible Navier-Stokes (NS) equations are often tested for code validation on a very well known benchmark problem; the lid-driven cavity flow. Due to the simplicity of the cavity geometry, applying a numerical method on this problem in terms of coding is quite easy and straight forward. Despite its simple geometry, the driven cavity flow retains a rich fluid flow physics manifested by multiple counter rotating recirculating regions on the corners of the cavity depending on the Reynolds number.[2]

This benchmark problem is selected in the present study as a good way to learn how to run a simulation in *Code_Saturne*, even though the generated laminar vortex does not provide significant data in the study of turbulence and the presence of boundary conditions introduce convective and pressure energy losses.

In the literature, it is possible to find different numerical approaches which have been applied to the lid-driven cavity flow problem. Though this problem has been numerically studied extensively, still there are some points which are no agreed upon. All these different approaches such as the famous Ghia *et al.*[29] are well summarized in Erturk *et al.*[2] so it serves as the reference benchmark data to this problem.

3.2.1 Problem Setup

The lid-driven cavity problem can be run using a variety of flow and initial conditions. The conditions used here are widely studied throughout the literature.

Flow conditions

A scheme of the test problem configuration is shown in Fig. 3.1 with the top wall moving to the right at a velocity V_0. The bottom and the two vertical walls are non-slip boundaries.

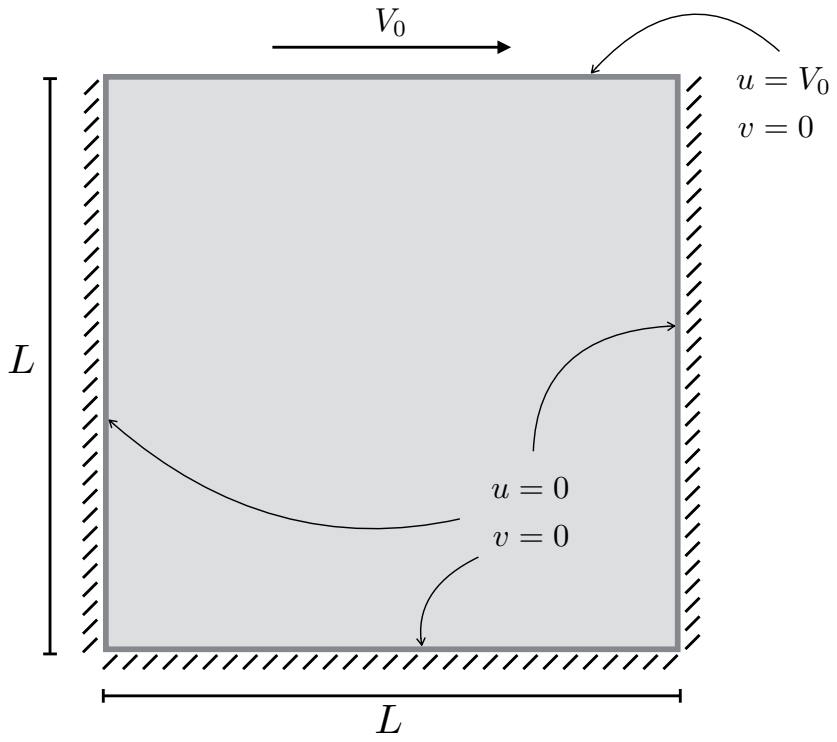


Figure 3.1: Schematic diagram for the lid-driven cavity flow problem with boundary conditions.

The Reynolds number selected for the test is 1000. Because *Code_Saturne* solves the equations in dimensional form additional flow conditions, given in Table 3.1, are selected to provide an incompressible flow at the given Reynolds number.

Table 3.1: Lid-driven cavity flow conditions.

Quantity	**Value**
Re	1000
V_0	1 m/s
L	1 m
ν	10^{-3} m^2/s

Numerical parameters

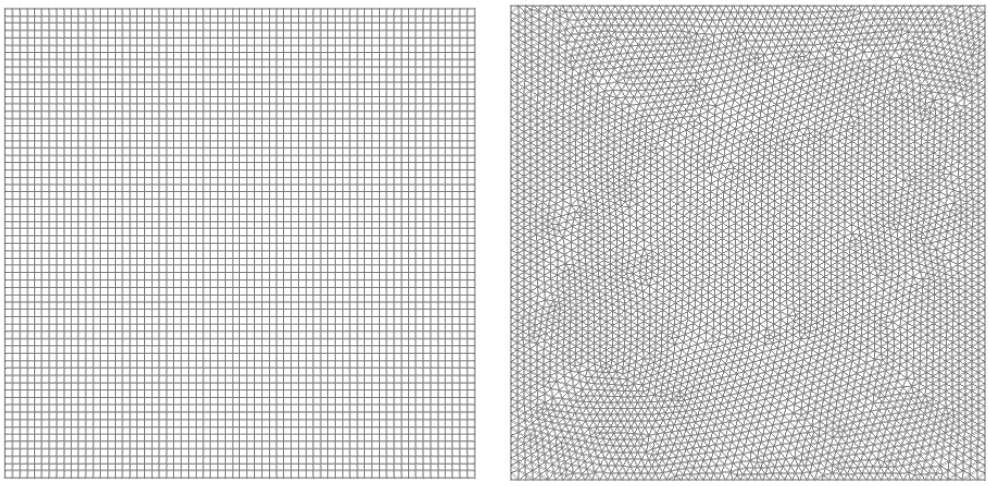
The problem is simulated in *Code_Saturne* under the ”Default” configuration. The parameters of this configuration are listed in Table 3.2. Refer to Appendix A for a further explanation of the meaning of each parameter.

Table 3.2: *Code_Saturne* ”Default” configuration.

Parameter	Value	Meaning
`arak`	1	Rhie and Chow interpolation activated
`imrgra`	0	Iterative reconstruction of the non-orthogonalities
`blencv`	1	Second order convective scheme
`ischcv`	1	Centered scheme
`isstpc`	0	Slope test activated
`pond`	-	Variable weighting factor
`ischtp`	1	First order time scheme
`nterup`	1	Single iteration in pressure-velocity coupling
`epsup`	10^{-5}	Relative precision in pressure-velocity coupling
`nswrsm(iu)`	1	Single iteration in velocity linear system
`nswrsm(ipr)`	2	Two iterations in pressure linear system
`epsilo(iu)`	10^{-8}	Relative precision in velocity linear system
`epsilo(ipr)`	10^{-8}	Relative precision in pressure linear system

Meshes

The meshes are generated through *ANSYS ICEM CFD* software. The meshes are three equally spaced cartesian grids of 64^2, 128^2, and 256^2 finite volumes (denoted with an ’s’) and a triangular non-structured mesh of 64^2 finite volumes.



(a) Structured 64^2s (b) Unstructured 64^2

Figure 3.2: 2D cavity mesh examples.

3.2.2 Results

The non-dimensional time step used for all the simulations is $\Delta t = 0.1$ and the tests are performed throughout 500 time steps, what corresponds to 50 units of simulation time.

All the cases are run on a single processor desktop computer with two cores. The elapsed time and the total CPU time for the runs is given in Table 3.3.

Table 3.3: Lid-driven cavity computing resources.

Mesh	Cores	Elapsed time (s)	Total CPU time (s)
64^2s	2	19.757	38.232
64^2	2	90.982	181.524
128^2s	2	113.731	227.252
256^2s	2	1032.315	2064.012

Baseline

The baseline set of simulations performed under the "Default" configuration provides the lid-driven cavity flow to converge to the steady state at approximately $t = 30$. Fig. 3.3 and Fig. 3.4 show the convergence of both u and v components of the velocity at four fixed points of the cavity.

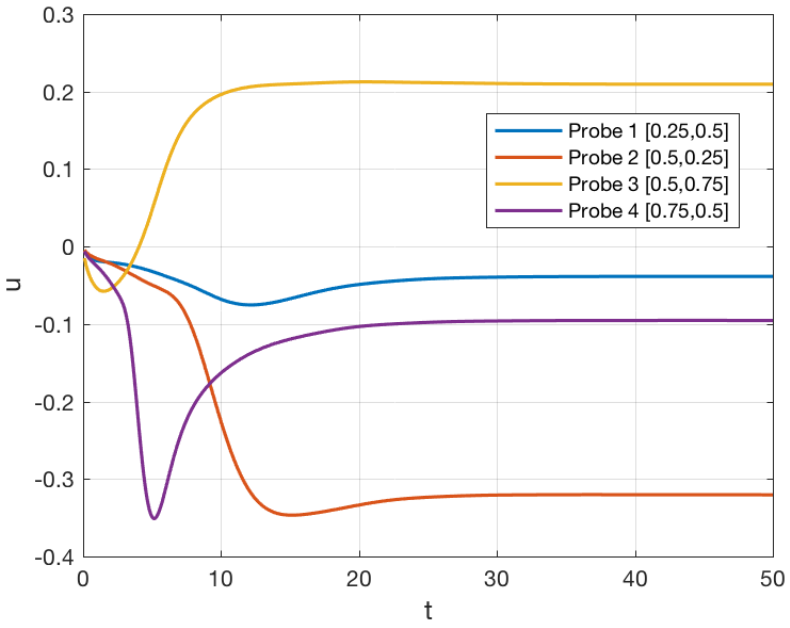


Figure 3.3: u-component velocity at four monitoring points as a function of time.

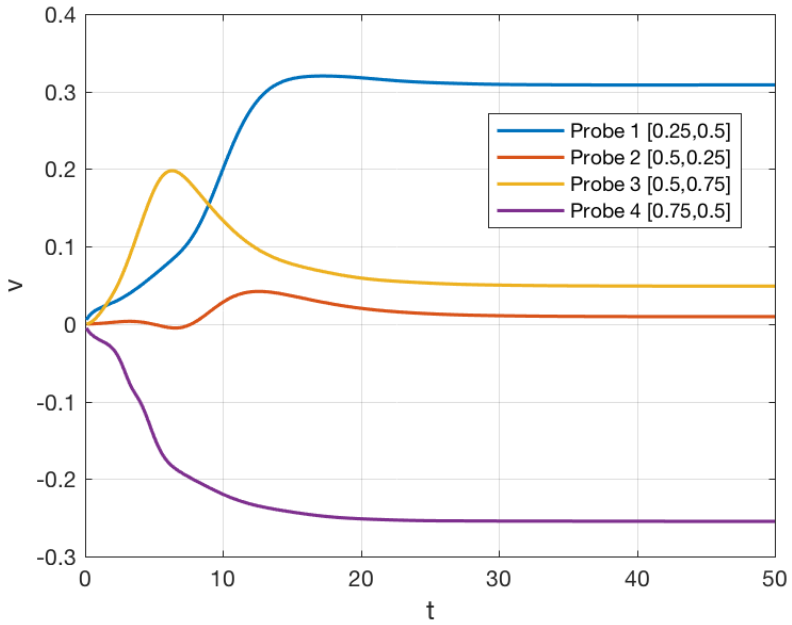


Figure 3.4: v-component velocity at four monitoring points as a function of time.

At the steady state, a primary clockwise vortex is generated and also two secondary counter clockwise vortex at the bottom corners of the cavity appear. Those are perfectly visible through the streamlines contours of the velocity field represented in Fig. 3.5.

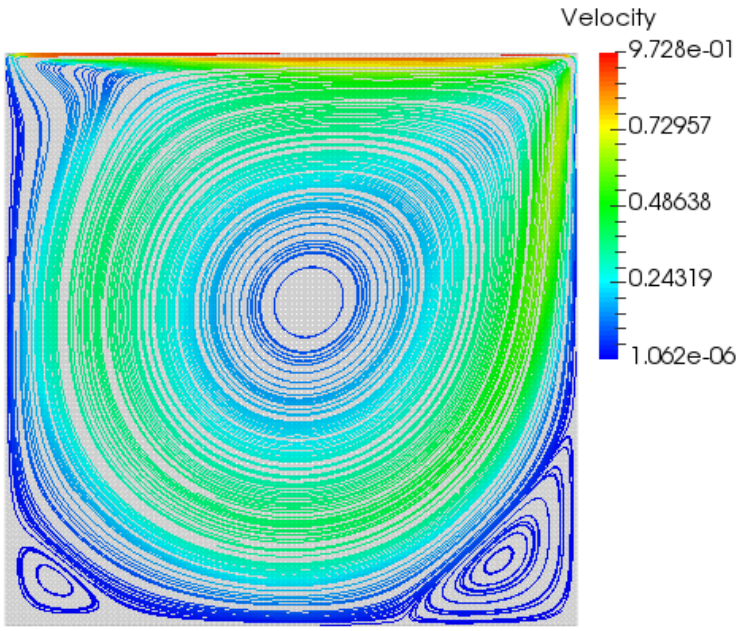
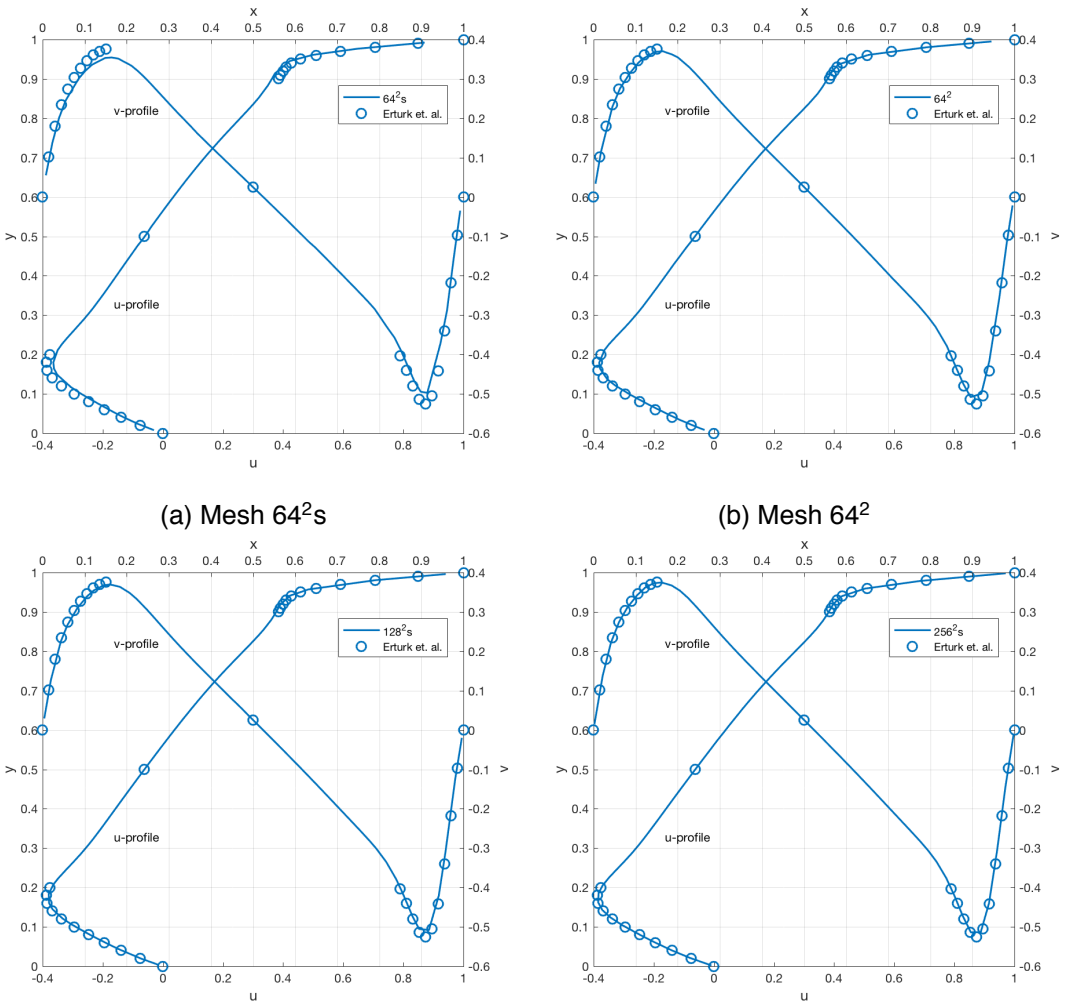


Figure 3.5: Streamline contours of primary and secondary vortices.

Mesh Refinement

The mesh refinement is carried in comparison to benchmark data. Erturk *et al.*[2] provides tabulated data of computed u and v profiles along a horizontal line passing through the geometric center of the cavity in a grid with 600^2 points. The results for each mesh size are shown in Fig. 3.6:



(a) Mesh 64^2s

(b) Mesh 64^2

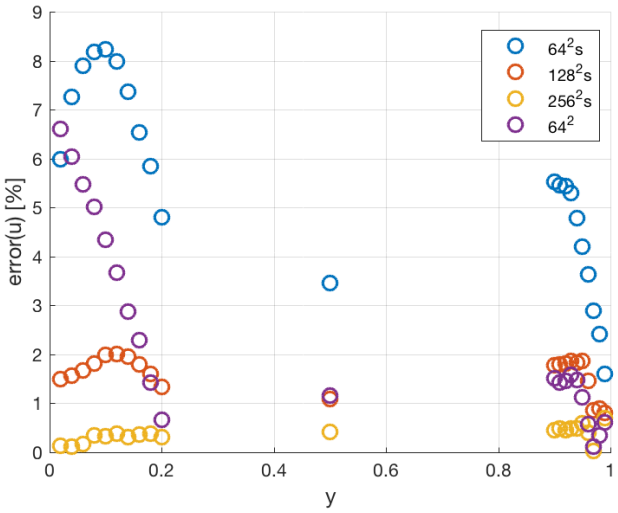
(c) Mesh 128^2s

(d) Mesh 256^2s

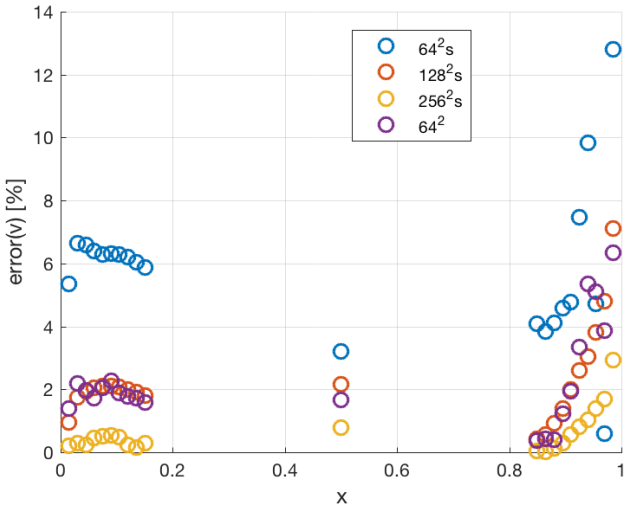
Figure 3.6: Velocity profiles comparison with Erturk *et al.*[2] at mid-sections.

At a first glance, the cartesian mesh with 64^2 points follows the correct profile but is far from the benchmark values. If the number of points is increased, the difference between the computed solution and the benchmark is decreased. This can be observed in cartesian grids for 128^2 and 256^2 points. The first one provides a closer solution without substantially increasing the computational time. On the other hand, the finest

grid needs eight times more CPU time but it shows almost the same results as those obtained with a 600^2 grid in Erturk *et al.*[2]. Thus, in this case, refining the mesh up to 256^2 is not worthy. Another interesting result is the behavior of the 64^2 unstructured mesh since it achieves nearly the same precision as the one obtained with a cartesian grid of 128^2 points. To get an more exact idea of those differences, the relative error to Erturk *et al.*[2] values has been computed in Fig. 3.7a and Fig. 3.7b.



(a) u-component



(b) v-component

Figure 3.7: Relative error of velocity profiles to Erturk *et al.*[2] for several types of mesh.

These plots confirm the above extracted conclusions, however, it is interesting to see how the unstructured mesh show a more irregular error pattern in comparison to the regular cartesian grids.

3.3 Two-Dimensional Flow: Taylor Vortex

The Taylor vortex flow is an exact 2D NS solution originally derived by G. I. Taylor in which the unsteady terms balance the diffusive terms, while the convective terms balance the pressure gradient. It can be considered a manufactured solution in the sense that it is a very particular case with no interest except from a theoretical point of view or for validation.

This benchmark problem is selected in the present study because its particular conditions provide an easy way to validate the energy conservation properties of a code. Therefore, the performance of *Code_Saturne* software is compared to a self-made spectro-consistent code in order to find which is the best configuration for *Code_Saturne* to perform as a conservative code.

Additionally, this benchmark test was used to test *Code_Saturne* in the aforementioned Dr. Benhamadouche's PhD thesis[23] for an inviscid case so it serves as the starting point of this study. However, in the following tests a more exhaustive approach is performed since the viscosity, hence the diffusive term, is taken into account. This fact is important due to the conclusions extracted in Section 2.3 where it is shown that the diffusive operator has to fulfill some conditions to preserve the kinetic energy.

Finally, this benchmark problem is the 2D version of the 3D Taylor-Green vortex that is studied in the following section. Thus, it is interesting to see how in 2D, the absence of vortex stretching affects to the decay of the vorticity in comparison to the fully turbulent 3D case.

3.3.1 Problem Setup

The Taylor vortex problem can be run using a variety of flow and initial conditions. The conditions used here are selected according to the 3D Taylor-Green vortex test case studied in Section 3.4.

Flow conditions

The domain consists of a periodic square 2D box in both x and y directions defined as $0 \leq x, y \leq L$. Within the domain an initial distribution of velocity and its corresponding vorticity is specified by the following relations.

$$u = V_0 \cos\left(\frac{2\pi x}{L}\right) \sin\left(\frac{2\pi y}{L}\right) \tag{3.5}$$

$$v = -V_0 \sin\left(\frac{2\pi x}{L}\right) \cos\left(\frac{2\pi y}{L}\right) \tag{3.6}$$

$$p = p_0 + \frac{\rho_0 V_0^2}{4}\left(\cos\left(\frac{4\pi x}{L}\right) + \cos\left(\frac{4\pi y}{L}\right)\right) \tag{3.7}$$

In the case of study, the flow is considered incompressible, therefore it is not needed to initialize the pressure as velocity and pressure fields are coupled through the Poisson equation as seen in Section 2.1.

As it has been already mentioned, this flow field has an analytical solution given by

$$u = V_0 \cos\left(\frac{2\pi x}{L}\right) \sin\left(\frac{2\pi y}{L}\right) F(t) \tag{3.8}$$

$$v = -V_0 \sin\left(\frac{2\pi x}{L}\right) \cos\left(\frac{2\pi y}{L}\right) F(t) \tag{3.9}$$

where F is

$$F(t) = e^{-8\pi^2 \nu t} \tag{3.10}$$

As expected, in the absence of body forces the total kinetic energy decays with time:

$$E_k(t) = \frac{F^2}{4} \tag{3.11}$$

The Reynolds number selected for the test is 1600. Because *Code_Saturne* solves the equations in dimensional form additional flow conditions, given in Table 3.4, are selected to provide an incompressible flow at the given Reynolds number.

Table 3.4: Taylor vortex flow conditions

Quantity	**Value**
Re	1600
V_0	1 m/s
L	1 m
ν	6.25 $\times$ 10^{-4} m^2/s

Numerical parameters

First of all, the problem is simulated in *Code_Saturne* under the ”Default” configuration (see Table 3.2). Then, a sensitivity analysis of the parameters is performed by means of testing several configurations and analyzing the kinetic energy conservation of each one. The several tested configurations and its acronyms are tabulated below.

Table 3.5: Test on convective scheme configurations.

Configuration	`blencv`	`arak`	`isstpc`	`pond`
Default	1	1	0	-
NoRhie	1	0	0	-
Full Upwind	0	1	0	-
Full Upwind + No Rhie	0	0	0	-
Centered + 0.2Upwind	0.8	1	0	-
Centered + 0.2Upwind + NoRhie	0.8	0	0	-
NoSlopeTest	1	1	1	-
NoSlopeTest + NoRhie	1	0	1	-
Pond 0.5	1	1	0	0.5
Pond 0.5 + NoSlopeTest	1	1	1	0.5

Table 3.6: Test on pressure-velocity coupling configurations.

Configuration	`nterup`	`epsilo`
Default	1	10^{-5}
Coupling1	10	10^{-5}
Coupling2	10	10^{-10}
Coupling3	100	10^{-5}

Table 3.7: Test on time scheme configurations.

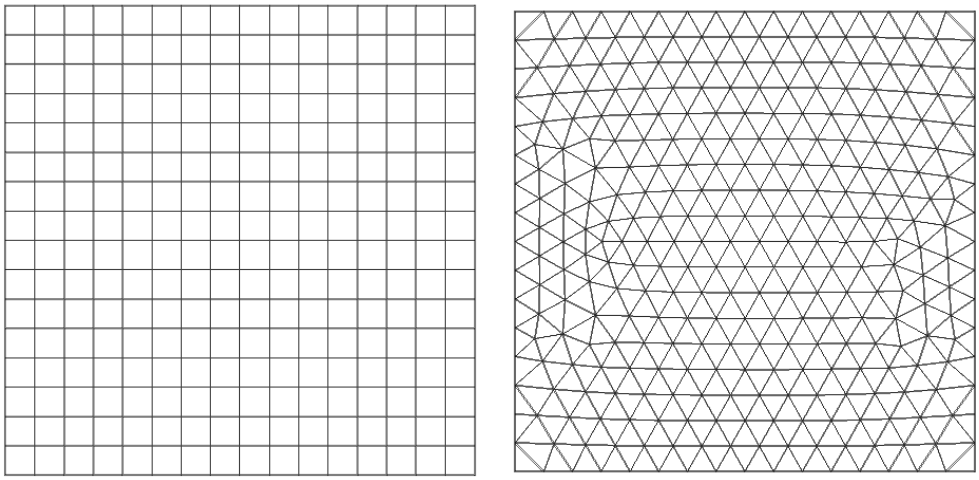
Configuration	`ischtp`	`nswrsm(iu,ipr)`	`epsilo(iu,ipr)`
Default	1	1, 2	10^{-8}, 10^{-8}
Time1	2	10, 5	10^{-5}, 10^{-5}
Time2	2	10, 5	10^{-10}, 10^{-10}
Time3	2	100, 100	10^{-5}, 10^{-5}

Table 3.8: Test on gradient calculation configurations.

Configuration	`imrgra`
Default	0
LeastSquare1	1
LeastSquare2	2
LeastSquare3	3
Iterative1	4
Iterative2	5
Iterative3	6

Meshes

The meshes are generated through *ANSYS ICEM CFD* software. The meshes are an equally spaced cartesian grids of 16^2 finite volumes (denoted with an 's') and a triangular non-structured mesh of 16^2 finite volumes.



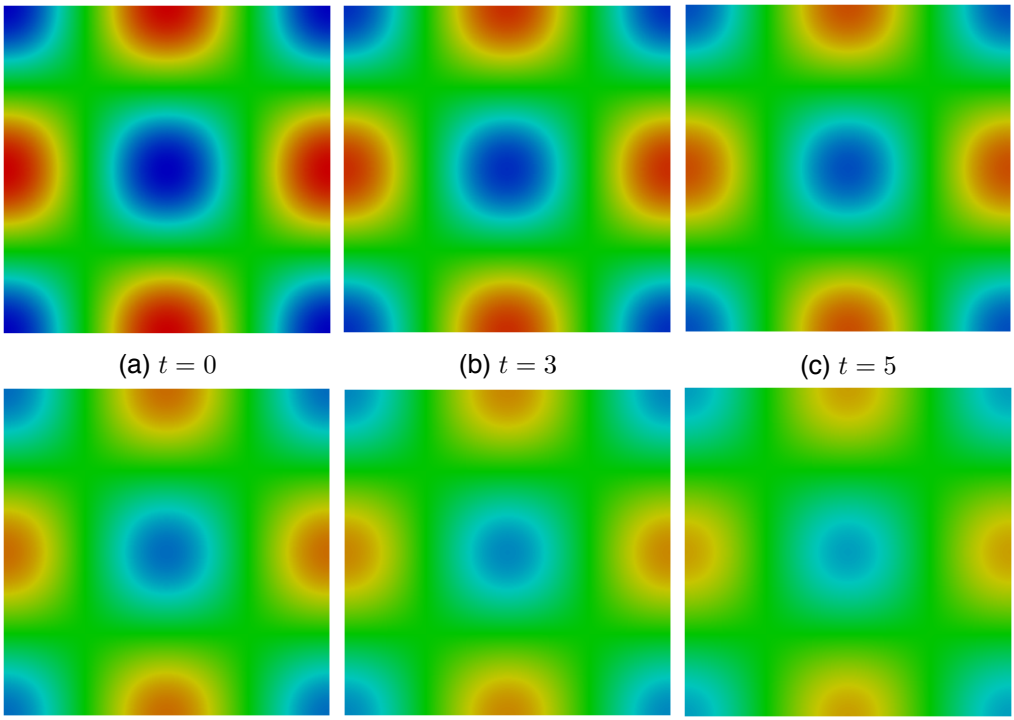
(a) Structured 16^2s (b) Unstructured 16^2

Figure 3.8: Taylor vortex 2D mesh examples.

3.3.2 Results

On the one hand, the problem is simulated in *Code_Saturne* with a constant non-dimensional time step of $\Delta t = 0.01$, which corresponds to a CFL of 0.16. The tests are performed along 2000 time steps, what corresponds to 20 units of simulation time. On the other hand, the problem is simulated in a self-made structured 2D spectro-consistent *Matlab* code following the symmetry-preserving discretization developed in Section 2.3. The non-dimensional time step is computed in each integration step through the CFL stability condition shown in Section 2.5.1. All the cases are run on a single processor desktop computer with two cores.

The vorticity evolution of the Taylor vortex is presented in Fig. 3.9. At $t = 0$ a periodic vorticity distribution is introduced and progressively decays with time due to viscous effects through the diffusive term. As it has been already mentioned, the absence of vortex stretching in 2D does not allow the vortex decay to break the vortices into turbulence and they remain as laminar vortices until they disappear up to $t = 11$ by viscous action.



(a) $t = 0$ (b) $t = 3$ (c) $t = 5$

(d) $t = 7$ (e) $t = 9$ (f) $t = 11$

Figure 3.9: z-vorticity evolution of the Taylor vortex.

Baseline

A baseline set of simulations is performed in *Code_Saturne* under the ”Default” configuration for both structured and unstructured meshes. Then, they are compared to the results from the spectro-consistent 2D code. The elapsed time and the total CPU time for the runs is given in Table 3.9. The spectro-consistent code shows to be slower than *Code_Saturne* software. The evolution of the kinetic energy is represented in Fig. 3.10.

Table 3.9: Taylor vortex initial computing resources.

Configuration	**Mesh**	**Cores**	**Elapsed time (s)**	**Total CPU time (s)**
Code_Saturne - Default	16^2s	2	3.096	6.136
Code_Saturne - Default	16^2	2	10.522	19.508
Spectro-consistent 2D code	16^2s	1	29.892	29.677

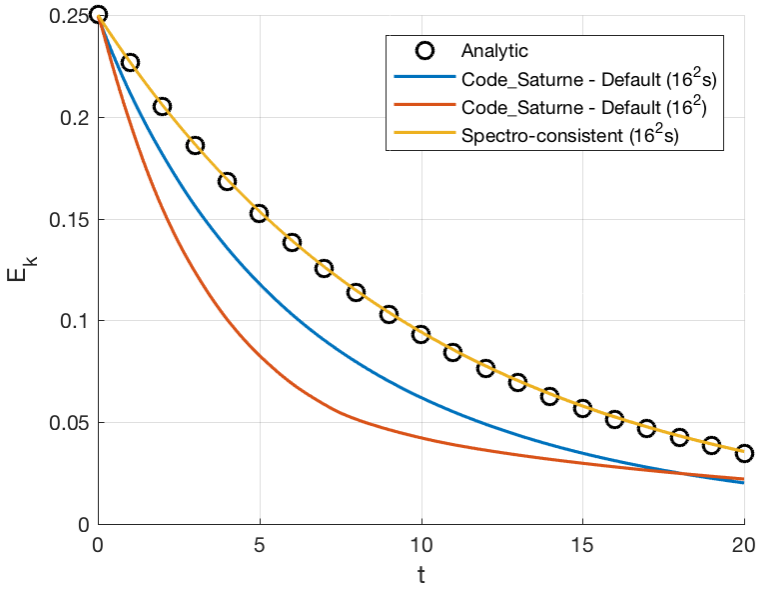


Figure 3.10: Evolution of kinetic energy by *Code_Saturne* "Default" configuration in comparison to the spectro-consistent 2D code.

The first results clearly show that the symmetry-preserving conditions of the spectro-consistent code allow to exactly preserve the kinetic energy on structured grids. However, under the same grid conditions, *Code_Saturne* "Default" configuration is dissipating an important amount of energy. Moreover, its behavior on an unstructured mesh is worse than on a structured mesh.

Tests on Regular Cartesian Mesh

In this section, the parameters of *Code_Saturne* are tested in order to find a more conservative configuration on structured meshes. From Dr. Benhamadouche's PhD thesis[23] it is known that on a regular Cartesian grid, the *Code_Saturne* unstructured formulation and the symmetry-preserving one should be equivalent.

First, a test on the convective scheme configuration is carried out. The several configurations of this test are explained in Table 3.5 and the kinetic energy results are shown in Fig. 3.11. For a regular cartesian mesh, it is not necessary to test `pond` weighting factor since it is geometrically 0.5.

In Fig. 3.11 it is easily illustrated that up-winding must be avoided. The numerical dissipation is dramatically increased with the full upwind scheme and is still very important

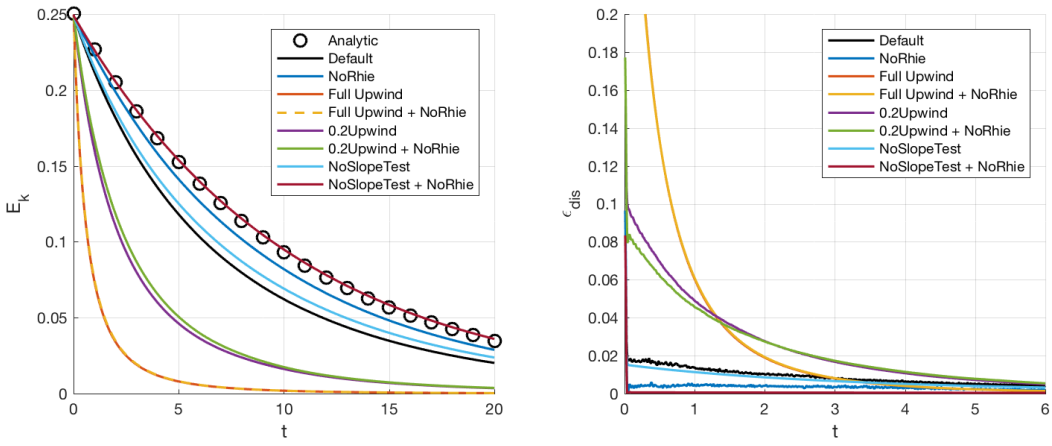
with the centered scheme with just 20% up-winding. The slope test in *Code_Saturne* (see Appendix A) switches locally to the upwind scheme wherever oscillations of the solution are detected, therefore its activation also introduces a numerical dissipation due to up-winding and it has to be deactivated.

All these cases are performed with or without Rhie and Chow interpolation (see Appendix A). Rhie and Chow interpolation has no perceptible effect with the full upwind scheme. The effect becomes more visible with the centered scheme blended with a low percentage of up-winding and becomes very important when a fully centered scheme is used.

In addition, it is found that with a fully centered scheme (second order in space), switching off both Rhie and Chow interpolation and the slope test allows *Code_Saturne* to preserve kinetic energy on regular Cartesian meshes (see "NoSlopeTest + NoRhie" in Fig. 3.11).

Table 3.10: Test on convective scheme for a structured mesh.

Configuration	**Cores**	**Elapsed time (s)**	**Total CPU time (s)**
Default	2	3.096	6.136
NoRhie	2	3.119	6.200
Full Upwind	2	2.777	5.520
Full Upwind + No Rhie	2	2.723	5.424
Centered + 0.2Upwind	2	3.205	6.432
Centered + 0.2Upwind + NoRhie	2	3.111	6.204
NoSlopeTest	2	2.710	5.436
NoSlopeTest + NoRhie	2	2.652	5.280



(a) Evolution of kinetic energy

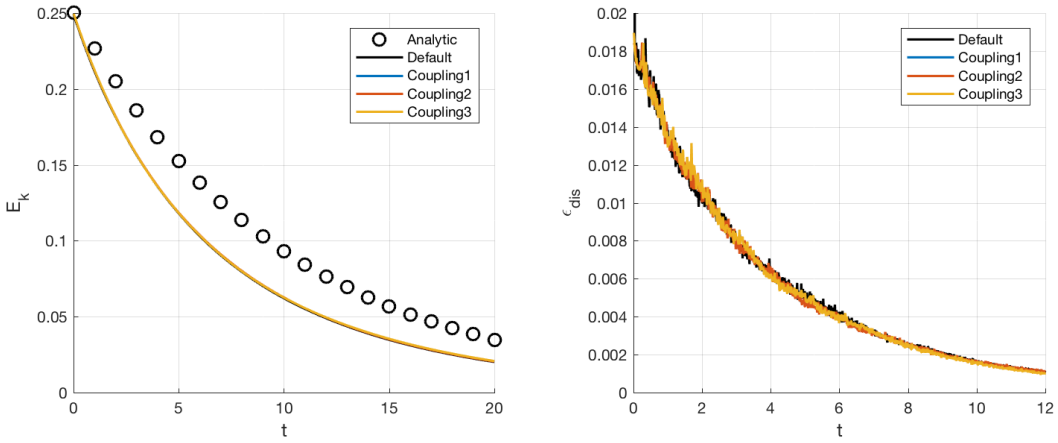
(b) Evolution of numerical dissipation

Figure 3.11: Comparison between several convective schemes on a structured mesh.

Next, the effects of the sub-iterations and the tolerance on pressure-velocity coupling are tested. In this case, Fig. 3.12 does not show any important differences between the several configurations (see Table 3.6). Nevertheless, the computational time that is required by each configuration is much more noticeable. ”Coupling1” and ”Coupling3”, with 10 and 100 sub-iterations, respectively, show that multiplying the number of sub-iterations by ten increases the CPU time by the same factor, whereas decreasing the tolerance from ”Coupling1” to ”Coupling2” has a minimal effect in terms of computational time. In Dr. Benhamadouche's PhD thesis[23] it is shown that a reasonable configuration to ensure a good balance between precision and CPU time in pressure-velocity coupling is ”Coupling1” with 10 sub-iterations and a tolerance of 10^{-5}.

Table 3.11: Test on pressure-velocity coupling for a structured mesh.

Configuration	**Cores**	**Elapsed time (s)**	**Total CPU time (s)**
Default	2	3.096	6.136
Coupling1	2	26.122	51.984
Coupling2	2	26.330	52.384
Coupling3	2	251.888	503.388



(a) Evolution of kinetic energy

(b) Evolution of numerical dissipation

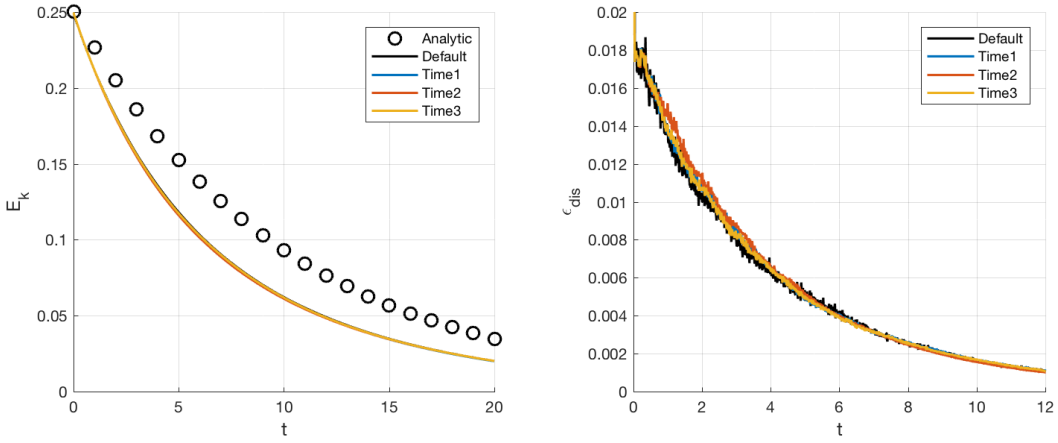
Figure 3.12: Comparison between several pressure-velocity coupling configurations on a structured mesh.

Now, the effects of the time scheme and its different configurations (see Table 3.7) are tested. Again, no differences can be noticed in Fig. 3.13. Even though moving from the first order time scheme of the ”Default” configuration to the second order time scheme of the ”Time1” configuration doubles the computational time, it is preferred a higher order scheme since then the error is quadratically reduced with the time step. Neither

reducing the tolerance (”Time2”) nor increasing the sub-iterations in the linear system (”Time3”) shows to be worthy in terms of computational time.

Table 3.12: Test on time scheme for a structured mesh.

Configuration	**Cores**	**Elapsed time (s)**	**Total CPU time (s)**
Default	2	3.096	6.136
Time1	2	6.391	12.588
Time2	2	7.908	15.616
TIme3	2	37.040	73.836



(a) Evolution of kinetic energy

(b) Evolution of numerical dissipation

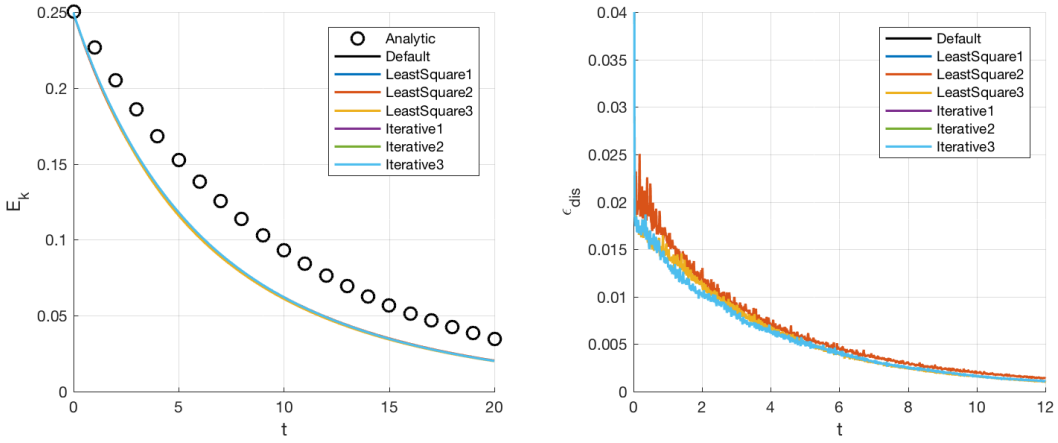
Figure 3.13: Comparison between several time scheme configurations on a structured mesh.

Finally, several methods to compute the gradients (see Table 3.8) are tested. In this case, no important differences can be noticed in Fig. 3.14 but the evolution of the numerical dissipation shows a slightly higher dissipation by the least squares method in comparison to the iterative ones. The most robust approaches are the iterative ones since the least squares method is quicker but does not guarantee the conservatively for the momentum equation. For these reasons, the ”Default” and ”Iterative2” configurations are preferred among the others. The most robust scheme is the second one (iterative reconstruction with initialization using least squares method based on the extended neighborhood) but the first (iterative reconstruction of the non-orthogonalities) is less CPU time demanding as seen in Table 3.13.

Taking into account all the above extracted conclusions, the most suitable configuration to be run on structured meshes by *Code_Saturne* is summarized in Table 3.14. It will be referred hereinafter under the name ”ConsStruct”.

Table 3.13: Test on gradient calculation for a structured mesh.

Configuration	Cores	Elapsed time (s)	Total CPU time (s)
Default	2	3.096	6.136
LeastSquare1	2	3.082	6.112
LeastSquare2	2	4.403	8.852
LeastSquare3	2	3.273	6.592
Iterative1	2	3.429	6.908
Iterative2	2	4.999	10.048
Iterative3	2	3.725	7.500



(a) Evolution of kinetic energy

(b) Evolution of numerical dissipation

Figure 3.14: Comparison between several gradient calculation schemes on a structured mesh.

Table 3.14: *Code_Saturne* "ConsStruct" configuration

Parameter	Value	Meaning
`arak`	0	Rhie and Chow interpolation deactivated
`imrgra`	5	Iterative reconstruction with initialization using least squares method based on the extended neighborhood
`blencv`	1	Second order convective scheme
`ischcv`	1	Centered scheme
`isstpc`	1	Slope test deactivated
`pond`	-	Variable weighting factor
`ischtp`	2	Second order time scheme
`nterup`	10	Ten iterations in pressure-velocity coupling
`epsup`	10^{-5}	Relative precision in pressure-velocity coupling
`nswrsm(iu)`	10	Ten iterations in velocity linear system
`nswrsm(ipr)`	5	Five iterations in pressure linear system
`epsilo(iu)`	10^{-5}	Relative precision in velocity linear system
`epsilo(ipr)`	10^{-5}	Relative precision in pressure linear system

Tests on Unstructured Mesh

In this section, other tests have been carried out with an unstructured grid containing tetrahedral elements. Here, only the parameters that can affect the unstructured formulation are tested. For this reason, the pressure-velocity coupling and the time schemes are not checked. Only the configurations that change the convective scheme are tested.

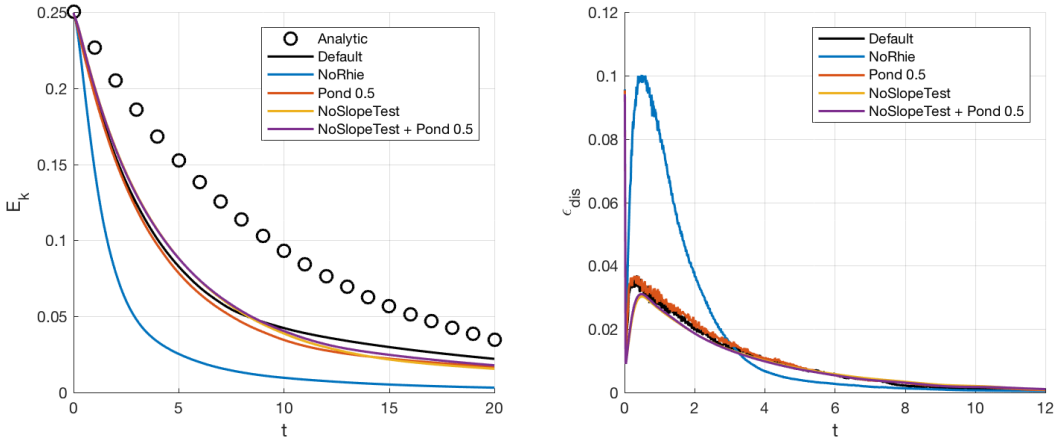
In unstructured grids, Rhie and Chow interpolation has undoubtedly a stabilizing effect. The symmetry-preserving formulation for the convection is not stable when the Rhie and Chow filter is removed. Although it has been proven that this configuration should conserve kinetic energy on regular Cartesian grids, it is obvious that it is not the case with unstructured meshes since the pressure term does not conserve energy and leads to instability (see ”NoRhie” in Fig. 3.15).

With the Rhie and Chow filter activated, the slope test and the weighting factor are tested. Fig. 3.15 shows that forcing the weighting factor to be 0.5 benefits the case without the slope test (full centered scheme), however, it is shown to be counter producing when some up-winding is activated due to the slope test.

Thus, the evolution of the kinetic energy and the numerical dissipation in Fig. 3.15 reveals that the ”NoSlopeTest + Pond 0.5” configuration performs better from $t = 0$ to $t = 8$. On the other hand, the ”Default” configuration turns into a better performance up to $t = 8$. Since the first one shows to be smoother where the energy is larger and preserves the symmetry in the convective scheme, it is considered as the preferred configuration.

Table 3.15: Test on convective scheme for an unstructured mesh.

Configuration	**Cores**	**Elapsed time (s)**	**Total CPU time (s)**
Default	2	10.522	19.508
NoRhie	2	9.433	18.624
Pond 0.5	2	9.800	19.324
NoSlopeTest	2	9.139	18.160
Pond 0.5 + NoSlopeTest	2	9.084	18.216



(a) Evolution of kinetic energy

(b) Evolution of numerical dissipation

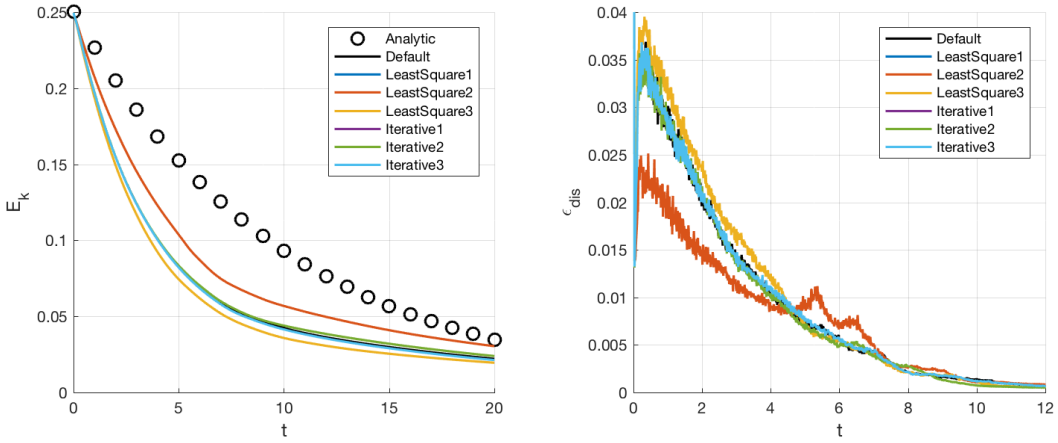
Figure 3.15: Comparison between several convective schemes on an unstructured mesh.

Next, the several gradient calculation schemes seen in Table 3.8 are tested again since their behavior should be different in an unstructured mesh because they have to deal with the non-orthogonalities of the mesh.

Indeed, the results shown in Fig. 3.16 show greater differences between schemes in comparison to the structured case and an important conclusion can be extracted. For non-structured tetrahedral grids the least squares method based on the extended neighborhood ("LeastSquare2" configuration) is the most robust approach and it significantly reduces the numerical dissipation even though it is CPU time demanding, as seen in Table 3.16. This conclusion is in accordance with the best practice guidelines of *Code_Saturne*[30].

Table 3.16: Test on gradient calculation for an unstructured mesh.

Configuration	**Cores**	**Elapsed time (s)**	**Total CPU time (s)**
Default	2	10.522	19.508
LeastSquare1	2	6.375	12.504
LeastSquare2	2	11.371	22.396
LeastSquare3	2	7.109	13.940
Iterative1	2	8.776	17.264
Iterative2	2	14.562	28.792
Iterative3	2	9.926	19.576



(a) Evolution of kinetic energy

(b) Evolution of numerical dissipation

Figure 3.16: Comparison between several gradient calculation schemes on an unstructured mesh.

Taking into account all the above extracted conclusions, the most suitable configuration to be run on unstructured meshes by *Code_Saturne* is summarized in Table 3.17. It will be referred hereinafter under the name ”ConsUnstruct”.

Table 3.17: *Code_Saturne* ”ConsUnstruct” configuration

Parameter	**Value**	**Meaning**
`arak`	1	Rhie and Chow interpolation activated
`imrgra`	2	Least squares method based on the extended neighborhood
`blencv`	1	Second order convective scheme
`ischcv`	1	Centered scheme
`isstpc`	1	Slope test deactivated
`pond`	0.5	Weighting factor forced to be 0.5
`ischtp`	2	Second order time scheme
`nterup`	10	Ten iterations in pressure-velocity coupling
`epsup`	10^{-5}	Relative precision in pressure-velocity coupling
`nswrsm(iu)`	10	Ten iterations in velocity linear system
`nswrsm(ipr)`	5	Five iterations in pressure linear system
`epsilo(iu)`	10^{-5}	Relative precision in velocity linear system
`epsilo(ipr)`	10^{-5}	Relative precision in pressure linear system

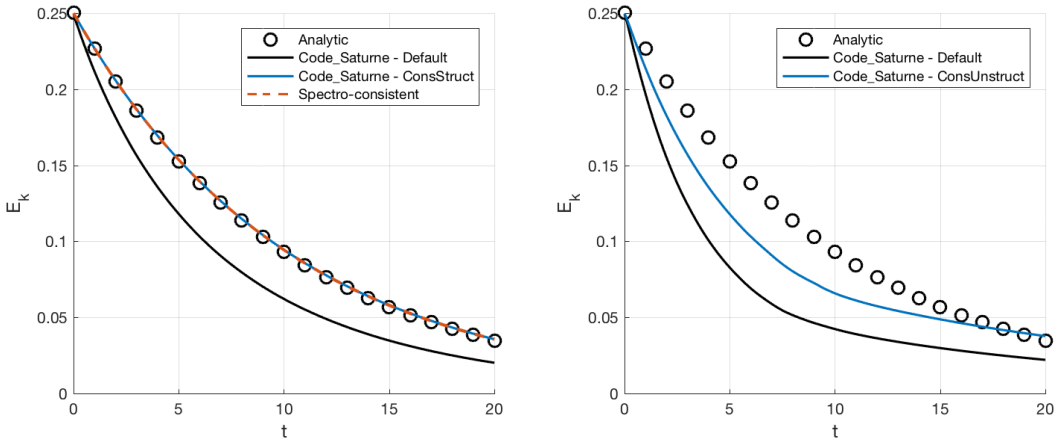
Conservative Performance

The last step is to perform a set of simulations in *Code_Saturne* under the conservative configurations extracted in the previous sections for both structured and unstructured meshes. The elapsed time and the total CPU time for the runs is given in Table 3.18. When it comes to computing performance, the *Code_Saturne* conservative configurations are more than two times slower than the default ones.

The evolution of the kinetic energy is represented in Fig. 3.17 for both structured and unstructured meshes of 16^2 points. With these results, it can be concluded that *Code_Saturne* unstructured formulation preserves kinetic energy on structured meshes if an appropriate configuration is chosen. However, on unstructured meshes, *Code_Saturne* does not conserve energy due to the pressure term, even though an improved configuration has been found to perform better than the configuration by default.

Table 3.18: Taylor vortex final computing resources.

Configuration	Mesh	Cores	Elapsed time (s)	Total CPU time (s)
Code_Saturne - Default	16^2s	2	3.096	6.136
Code_Saturne - ConsStruct	16^2s	2	8.465	16.976
Code_Saturne - Default	16^2	2	10.522	19.508
Code_Saturne - ConsUnstruct	16^2	2	24.536	47.928
Spectro-consistent 2D code	16^2s	1	29.892	29.677



(a) Structured 16^2s (b) Unstructured 16^2

Figure 3.17: Evolution of kinetic energy under the final configurations of *Code_Saturne* in comparison to the spectro-consistent 2D code.

3.4 Three-Dimensional Flow: Taylor-Green Vortex

The Taylor-Green Vortex (TGV) flow is a canonical problem in fluid dynamics developed to study vortex dynamics, turbulent transition, turbulent decay and the energy dissipation process. It is named after British physicist and mathematician G. I. Taylor and his collaborator A. E. Green due to their original work Taylor *et al.*[31] that provides an exact closed form solution of the incompressible NS equations in cartesian coordinates.

The TGV problem contains several key physical processes in turbulence in a simple construct and therefore is an excellent case for the evaluation of turbulent flow simulation methodologies. The problem consists of a cubic volume of fluid that contains a smooth initial distribution of vorticity. As time advances the vortices roll-up, stretch and interact, eventually breaking down into turbulence. Because there is no external forcing the small-scale turbulent motion will eventually dissipate all the energy in the fluid and it will come to rest.[32]

In the literature, it is possible to find different numerical approaches which have been applied to the TGV flow problem. An extensive analysis of this problem is performed by DeBonis[32] and Yilmaz *et al.*[33] employing high-resolution methods, and by Rees *et al.*[3] through a spectral method, which serve as reference benchmark data to this problem.

3.4.1 Problem Setup

The TGV problem can be run using a variety of flow and initial conditions. The conditions and post processing used here were specified by the organizers of the AIAA First International Workshop on High-Order Methods in Computational Fluid Dynamics[34].

Flow conditions

The domain consists of a periodic square box in all the three directions defined as $-\pi L \leq x, y, z \leq \pi L$. Within the domain an initial distribution of velocity and its corresponding vorticity is specified by the following relations. Again, in the case of study the flow is considered incompressible, therefore the initial pressure field is not needed (see Appendix C for mathematical proof).

$$u = V_0 \sin\left(\frac{x}{L}\right) \cos\left(\frac{y}{L}\right) \cos\left(\frac{z}{L}\right) \tag{3.12}$$

$$v = -V_0 \cos\left(\frac{x}{L}\right) \sin\left(\frac{y}{L}\right) \cos\left(\frac{z}{L}\right) \tag{3.13}$$

$$w = 0 \tag{3.14}$$

$$p = p_0 + \frac{\rho_0 V_0^2}{16}\left(\cos\left(\frac{2x}{L}\right) + \cos\left(\frac{2y}{L}\right)\right)\left(\cos\left(\frac{2z}{L}\right) + 2\right) \tag{3.15}$$

The Reynolds number selected for the test is 1600[34]. Because *Code_Saturne* solves the equations in dimensional form additional flow conditions, given in Table 3.19, are selected to provide an incompressible flow at the given Reynolds number.

Table 3.19: Taylor-Green vortex flow conditions

Quantity	Value
Re	1600
V_0	1 m/s
L	1 m
ν	6.25 $\times$ 10^{-4} m^2/s

Numerical parameters

The problem is simulated in *Code_Saturne* under the ”ConsStruct” configuration found in Section 3.3.2. The parameters of this configuration are listed in Table 3.14.

Meshes

The meshes are generated through *ANSYS ICEM CFD* software. The meshes are four equally spaced cartesian grids of 32^3, 64^3, 128^3 and 256^3 finite volumes.

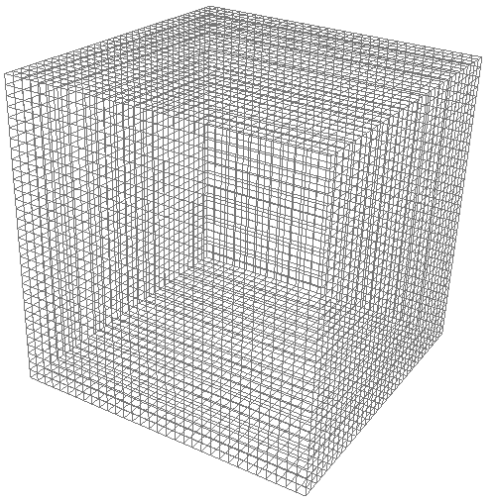


Figure 3.18: Taylor-Green mesh example of 32^3 finite volumes.

3.4.2 Results

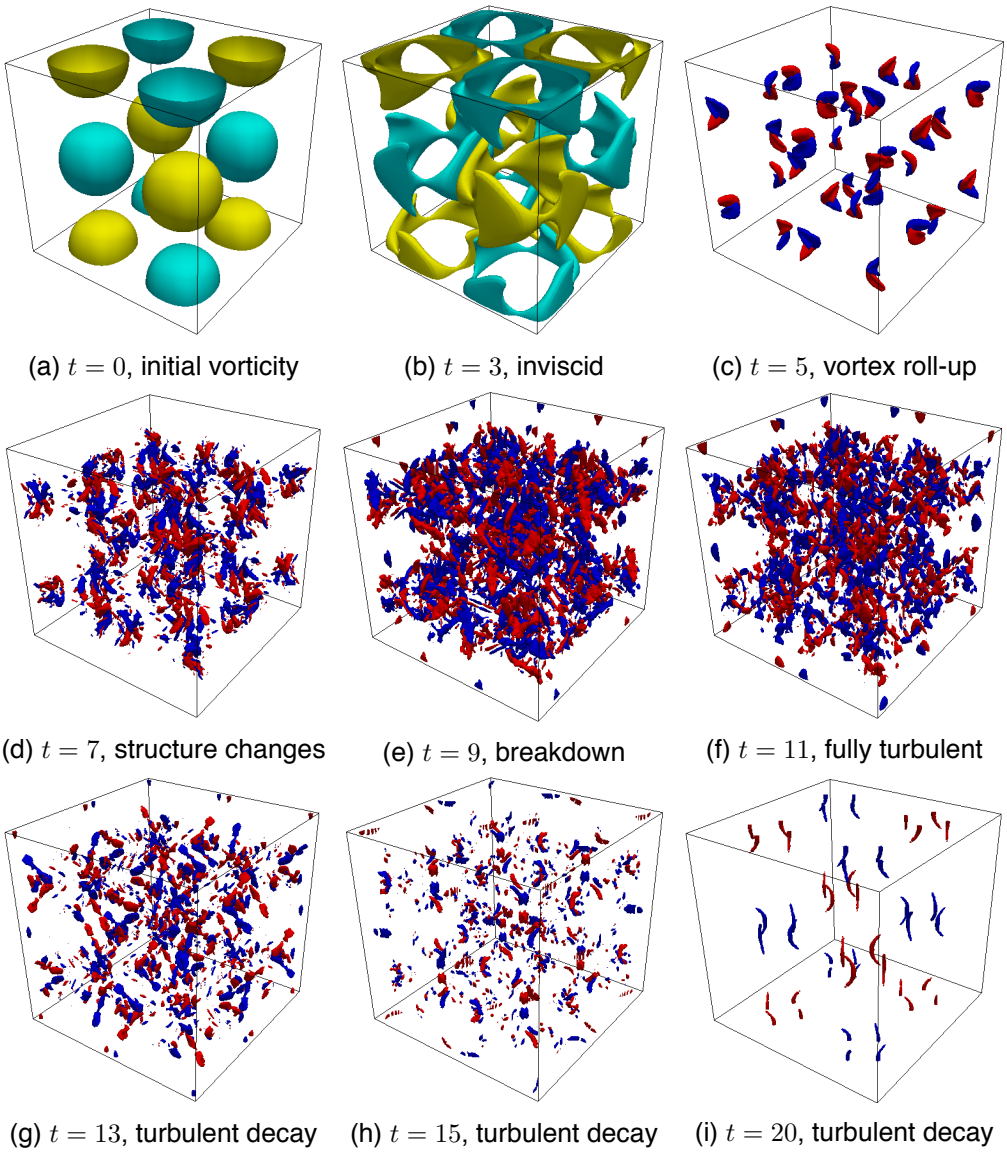
On the one hand, the problem is simulated in *Code_Saturne* selecting a non-dimensional time step for each grid size according to the stability study performed by DeBonis[32]. Henceforth, the non-dimensional time step used for the 32^3 case is $\Delta t = 6.770 \cdot 10^{-3}$ and the time step is halved for each doubling of the grid dimensions. On the other hand, the problem is also simulated in a 3D structured spectro-consistent code elaborated by the MSc candidate David Duran in his MSc thesis[35] following the symmetry-preserving discretization developed in Section 2.3. Due to the high computational demanding conditions of this 3D case, the cases are run thanks to the computing resources of the DFIS department. The elapsed time and the total CPU time for the runs is given in Table 3.20.

Table 3.20: Taylor-Green vortex computing resources.

Mesh	Cores	Elapsed time (s)	Total CPU time (s)
32^3s	2	1577.897	3154.260
64^3s	8	7230.633	57849.372
128^3s	8	118002.197	944057.364
256^3s	512	23794.996	12048550.25

Baseline

Iso-contours of the z-component of vorticity (Fig. 3.19) illustrate the evolution of the flow. At the earliest times the initially well organized flow behaves inviscidly as the vortices begin to evolve and roll-up. Near $t = 7$ the smooth vortical structures begin to lose their symmetries in time due to the energy cascade and the vortex stretching and start to break into smaller ones. Beyond this breakdown, the flow is fully turbulent and the structures slowly decay until the flow comes to rest.



(a) $t = 0$, initial vorticity (b) $t = 3$, inviscid (c) $t = 5$, vortex roll-up

(d) $t = 7$, structure changes (e) $t = 9$, breakdown (f) $t = 11$, fully turbulent

(g) $t = 13$, turbulent decay (h) $t = 15$, turbulent decay (i) $t = 20$, turbulent decay

Figure 3.19: Taylor-Green vortex iso-surfaces of z-vorticity (green and yellow: $\omega_z = \pm 1$; blue and red: $\omega_z = \pm 4$.

Mesh Refinement

The change in the kinetic energy over time is shown for all four grid levels in Fig. 3.20. Little difference is seen between the 64^3s, 128^3s and 256^3s grid levels, however, grid 32^3s is far from the reference solution that was obtained on a grid with 512^3 points.

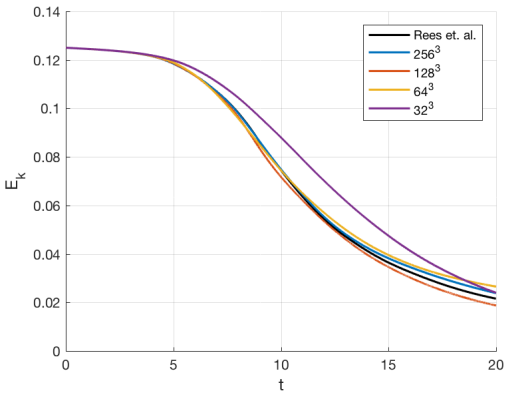
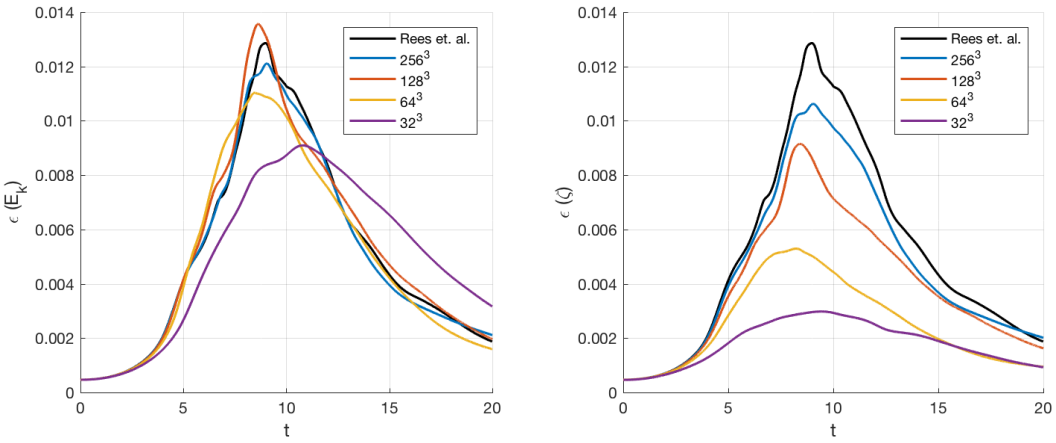


Figure 3.20: Kinetic energy comparison with Rees *et al.*[3] for several grids.

Fig. 3.21 shows the turbulent decay process. The directly computed kinetic energy dissipation rate shows again reasonable agreement with the reference solution for all grid levels unless for the 32^3s mesh. The largest discrepancies are at the peak dissipation rates, from $t = 7$ to $t = 12$. Nonetheless, the kinetic energy dissipation rate computed from the enstrophy should be equivalent to the directly computed one, but it is clear that there is a large discrepancy in the peak dissipation rate for the lower grid levels that improves with grid resolution.

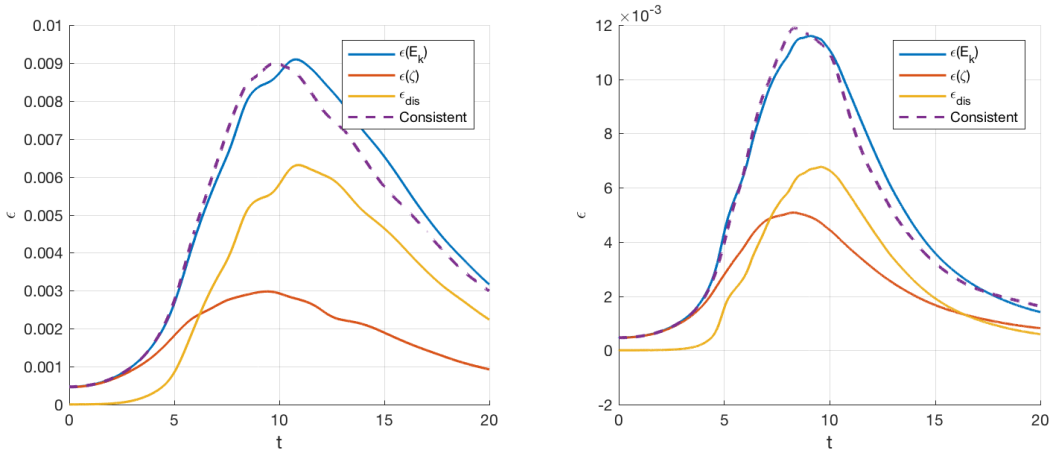


(a) Directly computed (b) Enstrophy based

Figure 3.21: Kinetic energy dissipation rate comparison for several grids.

Conservative Performance

Fig. 3.22 shows the results for each mesh compared to the results from the spectro-consistent 3D code. All the cases show that the directly computed kinetic energy dissipation rate (in blue) is in great agreement with the spectro-consistent solution. This means that *Code_Saturne* with the "ConsStruct" configuration is preserving kinetic energy, however, the discrepancies come from the enstrophy in the the peak dissipation rates. All the cases show no numerical dissipation (yellow line) from $t = 0$ to $t = 5$, while the vortices remain inviscid. When vortex stretching comes in and break the vortices into turbulence, the numerical dissipation is triggered due to 3D effects. Nevertheless, this numerical dissipation is shown to be reduced progressively with the number of points until the 256^3s mesh show great conservation properties.



(a) Mesh 32^3s

(b) Mesh 64^3s

(c) Mesh 128^3s

(d) Mesh 256^3s

Figure 3.22: Kinetic energy dissipation rate in comparison to spectro-consistent results (dashed line).

4

Project Management

In this chapter, a review of the initial planning is performed as well as the evaluation of the study in terms of economical and environmental feasibility.

4.1 Planning Review

At the beginning of the project, a charter was performed in order to plan and schedule the different tasks of the study according to its goals and scope. Within it, a Gantt diagram was developed and has been checked during the study realisation.

First of all, all the tasks have been performed as defined in the Project Charter. Moreover, the established order from the work break down structure has been respected. However, the time assigned to each task may differ from the established initial calendar since the author spent more time learning *Code_Saturne* software than it was initially expected. This delay affected the self-made code development since this task began later than planned, however, this delay did not prevent the successful completion of the task. The approximate time dedicated to each task has been computed and summarized in Table 4.1.

Table 4.1: Working hours.

Task	**h**
Literature research	30
Software learning	40
Preprocess	30
Code developing	100
Postprocess	50
Report	50
Total	**300**

In addition to the dedicated working hours, this study has required 200 hours of simulation hours between both *Code_Saturne* and self-made code. In this sense, it was part of the scope to properly manage this computational time constraints because it was the first time for the author to deal with such amount of simulations that could take up to 32 hours. Generally speaking, it has been satisfactorily managed but there have been some repetitive simulations that turned into erroneous results that could be avoided testing the problem in small parts instead of running the hole problem each time.

4.2 Economical Feasibility

During this study, the economical impact that has been taken into account is, on the one hand, due to working and simulation hours and, on the other hand, due to hardware and software amortization along the five months of duration of the study. The cost break down can be found in the Budget and the total cost of the study rises to 4590€.

4.3 Environmental Feasibility

During this project, the environmental impact that has been taken into account is due to the electricity costs associated to computer usage. However, since these activities are daily present in the life of the student regardless of the activity to do this project, a direct environmental impact of this project cannot be accounted and hence it is not addressed in terms of damage or pollution.

5

Conclusions and Future Work

The conclusions of this work are drawn in this chapter, as well as a discussion of the future developments. The aim of this work was to assess how *Code_Saturne* is able to characterize and solve turbulent flows.

The first part of this study was to learn the workflow of *Code_Saturne* and how to run simulations with this open-source software. During this first stage, the two dimensional lid-driven cavity benchmark test case was solved under the default configuration of *Code_Saturne*. In addition, a simple guide was elaborated to facilitate the initiation of any beginner to this software, as well as the summary of the numerical parameters that can be set with the meaning of each one and the recommended values from the software user's manuals.

Once the software was understood, the problem of energy conservation of the Navier-Stokes equations was tackled from a general point of view. An important state-of-the-art investigation about the symmetry-preserving conditions that a solver of the Navier-Stokes equations has to fulfill was carried in order to program a self-made two dimensional spectro-consistent structured code for structured meshes. These conditions were obtained analyzing the dissipation rate of the discrete kinetic energy, which showed that

in the absence of external sources, the overall contribution from convection and pressure gradient has to be zero for an incompressible flow, and that the diffusive term must be strictly dissipative. These conservation properties are held, if and only if, the discrete convective operator is skew-symmetric, the negative conjugate transpose of the discrete gradient operator is exactly equal to the divergence operator and the diffusive operator is symmetric and positive definite.

The developed software was verified through the 2D Taylor vortex obtaining excellent results. Then, this case was also used to perform an analysis of the parameters of *Code_Saturne*. Several important conclusions were found:

- The Rhie and Chow interpolation used in all the collocated approaches introduces a numerical dissipation that is more noticeable in centered convective schemes.
- A fully centered convective scheme can be obtained switching off the slope test.
- With a fully centered scheme, removing the Rhie and Chow interpolation for regular Cartesian grids allows to strictly conserve kinetic energy.
- Removing the Rhie and Chow interpolation leads to unstable solutions on fully unstructured meshes.
- Even with a symmetrical formulation on the convective term, the pressure gradient does not allow to conserve energy on unstructured grids.
- Forcing the weighting factor of the convective scheme to be 0.5 only benefits in a fully centered scheme on unstructured meshes.
- For structured meshes, the most robust gradient calculation approaches are the ones based on the iterative reconstruction of the non-orthogonalities, whereas for unstructured meshes, the most robust approach is the least squares method based on the extended neighborhood.

Taking all the above points into account, two parameter configurations for *Code_Saturne* were provided. The ”ConsStruct” configuration (see Table 3.14) that yields to preserve kinetic energy for structured meshes through a symmetrical formulation, and the ”ConsUnstruct” configuration (see Table 3.17) that provides a better performance than the default configuration, even though it does not exactly preserves kinetic energy.

Finally, the 3D Taylor-Green vortex was simulated under the conservative configuration. This fully turbulent case revealed that the presence of three dimensional effects due to vortex stretching causes numerical dissipation in *Code_Saturne*. The reason for this behavior is unknown for the author of the thesis since a deeper knowledge and analysis on the *Code_Saturne* collocated formulation is needed, however, it was shown that those discrepancies could be solved by refining the mesh size.

5.1 Future Actions

The future actions for this study include:

- Develop the extension of the two dimensional spectro-consistent code to three dimensions and work on the parallelization of the software to improve its performance on large scale systems. Some other test cases could be solved such as the channel flow.
- Investigate into the symmetry-preserving conditions that an unstructured code has to fulfill in order to avoid numerical dissipation on unstructured meshes due to the pressure gradient term, which has been shown to be a yet unsolved phenomena.
- Investigate into the formulation of *Code_Saturne* in order to find out why the vortex stretching three dimensional effects increase the numerical dissipation at low grid levels even though a conservative configuration for the convective term is used.
- Test all the several turbulence models available in *Code_Saturne* and extract conclusions about the capacity of each one to characterize and solve turbulent flows.
- Perform simulations of aerodynamic devices such an airfoil in order to study the ability and accuracy of *Code_Saturne* to compute valuable data about lift and drag forces, so then it can be used with confidence for aerodynamic simulation.

References

[1] Stephen B. Pope. *Turbulent Flows*. Cambridge University Press, 2000.

[2] E. Erturk, T. C. Corke, and C. Gökçöl. Numerical solutions of 2-D steady incompressible driven cavity flow at high Reynolds numbers. *International Journal for Numerical Methods in Fluids*, 48:747–774, 2005.

[3] W. M. van Rees, A. Leonard, D. I. Pullin, and P. Koumoutsakos. A comparison of vortex and pseudo-spectral methods for the simulation of periodic vortical flows at high Reynolds numbers. *Journal of Computational Physics*, 230:2794–2805, 2011.

[4] J.D. Anderson. *Computational Fluid Dynamics: The Basics with Applications*. McGraw-Hill, 1995.

[5] L. F. Richardson. Weather Prediction by Numerical Process. *Cambridge University Press*, 1922.

[6] Uriel Frisch. *The legacy of A.N. Kolmogorov*. Cambridge University Press, 1995.

[7] Joseph Smagorinsky. General Circulation Experiments with the Primitive Equations, Part I: The Basic Experiment. *Monthly weather review*, 91(3):99–152, 1963.

[8] F. Ducros, F. Nicoud, and T. Poinsot. Wall-Adapting Local Eddy-viscosity models for simulations in complex geometries. In *International Conference on Computational Conference*, pages 293–300, 1998.

[9] B.E. Launder and D.B. Spalding. The numerical computation of turbulent flows. *Computer methods in applied mechanics and engineering*, 3(2):269–289, 1974.

[10] F.R. Menter, M. Kuntz, and R. Langtry. Ten years of industrial experience with the SST turbulence model. *Turbulence, heat and mass transfer*, 4(1), 2003.

[11] P.R. Spalart and S.R. Allmaras. A one-equation turbulence model for aerodynamic flows. *AIAA*, 94, 1992.

[12] A. Arakawa. Computational Design for Long-Term Numerical Integration of the Equations of Fluid Motion: Two-Dimensional Incompressible Flow. Part 1. *Journal of Computational Physics*, 1(119-143), 1966.

[13] R. W. C. P. Verstappen and A. E. P. Veldman. Symmetry-preserving discretization of turbulent flow. *Journal of Computational Physics*, 187:343–368, 2003.

[14] R. W. C. P. Verstappen and A. E. P. Veldman. Spectro-consistent discretization of Navier-Stokes: a challenge to RANS and LES. *Journal of Engineering Mathematics*, 34(163-179), 1998.

[15] B. Perot. Conservation Properties of Unstructured Staggered Mesh Schemes. *Journal of Computational Physics*, 159:58–89, 2000.

[16] J. E. Hicken, F. E. Ham, J. Militzer, and M. Koksal. A shift transformation for fully conservative methods: turbulence simulation on complex, unstructured grids. *Journal of Computational Physics*, 208:704–734, 2005.

[17] F. X. Trias, O. Lehmkuhl, A. Oliva, C. D. Pérez-Segarra, and R. W. C. P. Verstappen. Symmetry-preserving discretization of Navier-Stokes equations on collocated unstructured grids. *Journal of Computational Physics*, 258:246–267, 2014.

[18] EDF. Code_saturne version 4.0.0 installation guide, 2015.

[19] Y. Fournier, J. Bonelle, Z. Shang, A. G. Sunderland, and J. C. Uribe. Optimizing *Code_Saturne* computations on Petascale systems. *Computers and Fluids*, 45: 103–108, 2011.

[20] Partnership for Advanced Computing in Europe. URL `http://www.prace-ri.eu`.

[21] F. Archambeau, C. Béchaud, B. Gest, A. Martin, and M. Sakiz. Qualification of *Code_Saturne* for thermal-hydraulics single phase nuclear applications. Technical report, EDF R&D, Fluid Mechanics and Heat Transfer Division, 2001.

[22] F. Archambeau, N. Méchitoua, and M. Sakiz. *Code Saturne: A Finite Volume Code for the computation of turbulent incompressible flows - Industrial Applications*, volume 1. 2004.

[23] S. Benhamadouche. *Large Eddy Simulation with the unstructured collocated arrangement*. PhD thesis, University of Manchester, 2006.

[24] C. M. Rhie and W. L. Chow. A numerical study of the turbulent flow past an isolated airfoil with training edge separation. *AIAA paper*, 82-0998, 1982.

[25] F. X. Trias. *Direct numerical simulation and regularization modelling of turbulent flows on loosely coupled parallel computers using symmetry-preserving discretizations*. PhD thesis, Universitat Politècnica de Catalunya, 2006.

[26] F. H. Harlow and J. E. Welch. Numerical Calculation of Time-Dependent Viscous Incompressible Flow of Fluid with Free Surface. *Physics of Fluids*, 8(2182-2189), 1965.

[27] A. J. Chorin. Numerical Solution of the Navier-Stokes Equations. *Journal of Computational Physics*, 22:745–762, 1968.

[28] R. Courant, K. Friedichs, and H. Lewy. On the Partial Difference Equations of Mathematical Physics. *IBM Journal*, 1967.

[29] U. Ghia, K. N. Ghia, and C. T. Shin. High-Re Solutions for Incompressible Flow Using the Navier-Stokes Equations and Multigrid Method. *Journal of Computational Physics*, 48:387–411, 1982.

[30] EDF. Code_Saturne Best Practice Guidelines in Numerical Parameters, 2015.

[31] G.I. Taylor and A.E. Green. Mechanism of the Production of Small Eddies from Large Ones. *Proceedings of the Royal Society of London*, 158:499–521, 1936.

[32] James R. DeBonis. Solutions of the Taylor-Green Vortex Problem Using High-Resolution Explicit Finite Difference Methods. Technical Report 217850, National Aeronautics and Space Administration, 2013.

[33] I. Yilmaz and L. Davidson. Comparison of SGS models in LES for transition to turbulence in Taylor-Green flow. *Conference on modelling fluid flow*, 2015.

[34] AIAA. Problem C3.5 Direct Numerical Simulation of the Taylor-Green Vortex at Re = 1600. In *1st International Workshop on High-Order CFD Methods*, 2012.

[35] David Duran. Study of the boundary layer flow control using synthetic jets by means of spectro-consistent discretizations. Master's thesis, UPC, 2017.